Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

Sampsa Latvala

Evaluation of Out-of-Band Authentication Channels

Master's Thesis
Espoo, May 27, 2019

Supervisor: Professor Tuomas Aura
Advisor: Mohit Sethi D.Sc. (Tech.)

Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

ABSTRACT OF MASTER'S THESIS

Author: Sampsa Latvala

Title:
Evaluation of Out-of-Band Authentication Channels

Date: May 27, 2019 **Pages:** viii + 71

Major: Computer Science **Code:** SCI3042

Supervisor: Professor Tuomas Aura

Advisor: Mohit Sethi D.Sc. (Tech.)

One of the challenges in entirely wireless communication systems is authentication. In pervasive computing and peer-to-peer networks, it is often not possible to rely on the existence of a trusted third party or other infrastructure. Therefore, ad hoc verification of keys via an out-of-band (OOB) channel is often the only way to achieve authentication.

Nimble out-of-band for EAP (EAP-NOOB) protocol is intended for bootstrapping security between IoT devices with no provisioned authentication credentials and minimal user interface. The protocol supports a user-assisted OOB channel to mutually authenticate the key-exchange performed over an insecure wireless network between the peer and the server. The protocol allows peers to scan for available networks and, based on the results, generate multiple dynamic OOB messages. The user then delivers one of these messages to the server to register the device and authenticate the key-exchange.

We implemented the OOB channels using NFC, QR codes and sound with EAP-NOOB as the bootstrapping protocol. The implementation requires an auxiliary device such as the user's smartphone. We evaluated the usability and security as well as the benefits and limitations of the OOB channels.

Our results show that NFC and QR codes are capable in displaying multiple OOB messages while the sound-based channel is suitable for one or two messages due to its lower bandwidth. When the peer device generates multiple OOB messages, the process becomes more complex for the user who needs to browse through them and identify the correct server. However, we showed that this cumbersome step can be removed with the help of a mobile application. Furthermore, we identified vulnerabilities in each technology when used as an OOB channel. While some of these vulnerabilities can be mitigated with the mobile application, some require more refined solutions.

Keywords: Authentication, EAP-NOOB, Out-of-band authentication, Near-Field Communication, Quick Response code, IoT, Chirp

Language: English

Aalto-yliopisto
Perustieteiden korkeakoulu
Tieto-, tietoliikenne- ja informaatiotekniikan maisteriohjelma

DIPLOMITYÖN TIIVISTELMÄ

Tekijä: Sampsa Latvala

Työn nimi:
Kaistan ulkopuolisten todennuskanavien arviointi

Päiväys:	27. toukokuuta 2019	**Sivumäärä:**	viii + 71
Pääaine:	Tietotekniikka	**Koodi:**	SCI3042
Valvoja:	Professori Tuomas Aura		
Ohjaaja:	TkT Mohit Sethi		

Yksi täysin langattomien järjestelmien haasteista on todennus. Sulautetussa tietotekniikassa sekä vertaisverkkoissa ei usein voida luottaa maailmanlaajuisesti luotettavan kolmannen osapuolen olemassaoloon. Siksi salausavainten ad hoc-varmennus erillistä tiedonsiirtokanavaa (OOB) käyttäen on usein ainoa ratkaisu turvallisen kommunikaation käynnistämiseksi. Se luo resilienssiä eri hyökkäyksiä vastaan tuomalla järjestelmään toisen, itsenäisen tiedonsiirtokanavan.

EAP-NOOB protokolla on tarkoitettu IoT-laitteille, joilla on minimaalinen käyttöliittymä eikä esiasennettuja avaimia. EAP-NOOB tukee käyttäjäavustettua OOB-tiedonsiirtokanavaa, jota käytetään todentamaan suojaamattomassa verkossa suoritettu laitteen ja palvelimen keskinäinen salausavainten vaihto. Protokolla sallii laitteiden kartoittaa käytettävissä olevia verkkoja ja tuottaa sen perusteella dynaamisia todennusviestejä, jotka käyttäjä toimittaa palvelimelle laitteen rekisteröimiseksi.

Tässä työssä tutkittiin EAP-NOOB protokollan OOB kanavaa käyttäen NFC:tä, QR-koodeja ja ääntä. Todennusviestin lukeminen laitteelta vaatii käyttäjältä älypuhelimen. Työssä arvioitiin toteutettujen todennuskanavien käytettävyyttä, tietoturvaa, hyötyjä sekä näitä rajoittavia tekijöitä.

Työn tulokset osoittavat, että NFC ja QR-koodit soveltuvat näyttämään useita OOB-viestejä. Sen sijaan äänipohjainen kanava soveltuu vain yhdelle tai kahdelle viestille hitaamman tiedonsiirron johdosta. Kun IoT-laite tuottaa useita OOB-viestejä, käyttäjäkokemus muuttuu monimutkaisemmaksi, koska käyttäjän on tunnistettava oikea viesti ja palvelin. Työssä osoitetaan, että tämä käyttäjälle hankala vaihe voidaan välttää erillisellä mobiilisovelluksella. Lisäksi työssä tunnistettiin toteutettujen tiedonsiirtomenetelmien haavoittuvuuksia, kun niitä käytettiin OOB-kanavana. Vaikka osa näistä haavoittuvuuksista voidaan eliminoida mobiilisovelluksen avulla, jotkut niistä vaativat tehokkaampia ratkaisuja.

Asiasanat:	Todennus, EAP-NOOB, kaistan ulkopuolinen todentaminen, NFC-lähitiedonsiirtoteknologia, QR-koodi, Esineiden internet, Chirp
Kieli:	Englanti

Acknowledgements

First and foremost, I would like thank Professor Tuomas Aura for his time, feedback and valuable suggestions during the thesis work. I would also like express my sincere gratitude to D.Sc. Mohit Sethi for his support and for providing feedback on my writing. Finally, this thesis would not have been possible without the endless support from my family.

Espoo, May 27, 2019

Sampsa Latvala

Abbreviations and Acronyms

AAR	Android Application Record
AP	Access Point
APDU	Application Protocol Data Unit
API	Application Programming Interface
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
DH	Diffie-Hellman key exchange
DRM	Digital Rights Management
EAP	Extensible Authentication Protocol
HISP	Human Interactive Security Protocol
Hoob	Cryptographic fingerprint used in OOB messages
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	Identifier
IDE	Integrated Development Environment
IoT	Internet of Things
ISO	International Organization for Standardization
MAC	Message Authentication Code
MIME	Multipurpose Internet Mail Extensions
MITM	Man-in-the-Middle attack
MMI	Man-Machine Interface
NDEF	NFC Data Exchange Format
NFC	Near-Field Communication
NOOB	Nimble Out-of-Band
Noob	Secret nonce used in OOB messages
OOB	Out-of-band
OTP	One-time Password

PIN	Personal Identification Number
PPI	Pixels Per Inch
QR	Quick Response
RFC	Request For Comments
RFID	Radio-Frequency Identification
RTD	Record Type Definition
SDK	Software Development Kit
SMS	Short Message Service
SNEP	Simple NDEF Exchange Protocol
TLS	Transport Layer Security
TNF	Type Name Field
TTL	Time to Live
TUPAS	Tunnistuspalvelu Standardi; Digital identification service
UI	User Interface
UID	User Identifier
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VLC	Visual Light Communication

Contents

Chapter 1

Introduction

The number of Internet of Things (IoT) devices is growing rapidly. According to a forecast [17], by 2022 the amount of Internet connected devices will reach about 29 billion, of which 18 billion will be IoT related devices. IoT devices include connected cards, machines, instruments, wearables and other consumer electronics. Network connectivity aims to improve user experience by automating various processes and exchanging information without user input. While IoT growth provides great benefits and opportunities, there exist severe security risks [58]. Insecure IoT devices may act as gateways for attacks against the entire Internet infrastructure. According to Oracevic et al. [58], the biggest challenges in IoT are presently related to data and privacy protection. Therefore, it is important that the devices have robust security mechanisms, such as secure device bootstrapping. Secure device bootstrapping refers to the process of establishing a secure connection between two previously unassociated devices.

According to Fischer et al. [22], reliable device authentication is important in IoT and open unattended environments as it is the base for many security mechanisms, such as authorization, integrity checks, and secure configuration. Currently, there exists no agreed security standards for IoT manufacturers to take advantage of while developing and manufacturing their products [81]. Therefore, manufacturers use a variety of proprietary solutions — each taking their own approach. Most commercial IoT devices rely on the custom security and authentication solutions. This contradicts the idea of seamless interoperability of the devices in the Internet of Things (IoT) while also adding costs and complexity to the manufacturing process. Furthermore, users are often confused when different devices require different

processes for configuration.

Extensible Authentication Protocol (EAP) is an authentication framework and supports numerous methods for authentication [79]. Nimble out-of-band for EAP (EAP-NOOB) is a new EAP method and provides a nimble approach for device bootstrapping. EAP-NOOB is an authentication protocol intended specifically for bootstrapping various IoT devices with limited input and output capabilities [7]. Compared to most other EAP methods, EAP-NOOB does not require any pre-configured credentials. Instead, device configuration and registration to a server database along with ownership information and newly created authentication credentials are performed during the initial device deployment. For this, the protocol takes advantage of a user assisted out-of-band (OOB) channel for verifying the security of key-exchange, which takes place on the in-band channel.

1.1 Research goals and methodology

This thesis begins by performing a literature survey of OOB security protocols and existing devices that use OOB channels. The thesis then provides a summary of the EAP-NOOB protocol. Using EAP-NOOB bootstrapping protocol, we evaluate a variety of different OOB channels. In particular, we examine Near-Field Communication (NFC), audio and Quick Response (QR) codes. We evaluate the benefits and limitations of each OOB channel against the unique requirements of the EAP-NOOB protocol. In addition, an Android application is developed to examine the security and user experience of the paring process with the EAP-NOOB protocol.

To summarize, the research goals of this thesis are as follows:

1. A thorough literature survey of the use of OOB channels in security protocols.

2. Document the EAP-NOOB protocol and its unique OOB requirements.

3. Implement three different OOB channels for the EAP-NOOB protocol.

4. Analyze the benefits and limitations of the three OOB channels with EAP-NOOB as the bootstrapping protocol.

1.2 Structure of the thesis

The rest of the thesis is structured as follows: Chapter 2 describes the background of the current work, including some existing standards and products that rely on OOB channels. Chapter 3 provides an overview of the EAP-NOOB protocol. Chapter 4 presents the technologies which we evaluate for the OOB channel. Thereafter, in Chapter 5, we describe proof-of-concept implementations of the OOB channels. Chapter 6 evaluates each of the technologies from the usability and security perspectives. New insights and findings are discussed in Chapter 7. Finally, Chapter 8 concludes this thesis.

Chapter 2

Background

In this chapter we discuss authentication methods that rely on OOB channels. We first look at how OOB channels are typically used in security protocols. Thereafter, we explain why OOB channels are needed for IoT devices. Furthermore, we describe commercial implementations and standards relying on OOB channels for the security bootstrapping.

2.1 Out-of-band

Out-of-band (OOB) refers to a separate communication channel severed from the primary in-band channel over which the actual network communication occurs [14, 37]. If a system takes advantage of OOB authentication, there must exist a separate communication medium used for authenticating an entity in the system. This secondary channel is often either used to transmit an authenticated shared secret or to verify information passed over the primary communication channel. The data exchanged in the OOB channel can vary from a hash value determined from the endpoint's public keys to the transcript of a key-exchange performed on the in-band channel [50, 65]. There are multiple ways of taking advantage of the OOB channel. The OOB methods can be divided roughly into four different classes based on their characteristics.

1. **Direct key-provisioning.** In this method, the cryptographic keys are provisioned directly over the OOB channel. Currently, the security strength of at least 128 bits is necessary for establishing cryptographic protection. Therefore, this approach requires a relatively long OOB

message for satisfying the current cryptographic key length requirement. For stronger keys, such as 256 or 368 bit the message becomes even longer and, therefore, the channel bandwidth plays an important role. Each key update requires a repeated OOB communication. Furthermore, when the cryptographic keys are provisioned directly over the OOB channel, the confidentiality of the channel becomes extremely important.

2. **Confirming key-exchange.** In this method the OOB channel is utilized for only verifying the Diffie-Hellman (DH) key exchange performed over the in-band channel. Therefore, the the OOB messages are often short.

3. **Fuzzy OOB channels.** In this method, the devices are paired using an error-prone OOB channel or a secret, which is transferred over a lossy analogue channel. The fuzzy secret may, for example, be an ambient sound or simultaneous user action, such as synchronized drawing [66]. This is typically used to confirm the key exchange performed over the primary channel.

4. **OOB channel for login services.** In this type of OOB method, the user is provided with an additional secret, e.g., a PIN code via SMS or a password via email. This approach provides the user with additional information required in the login process in addition to the traditional login credentials, such as user ID and password.

2.1.1 Need for OOB authentication in IoT

According to Mayrhofer et al. [48], a major problem in entirely wireless communication systems is the key management and authentication, e.g., how to securely exchange keys with the right entity. Taking advantage of secure key exchange protocols, such as Diffie-Hellman, highlights the problem of authentication, i.e., problem of verifying that the key truly belongs to the right entity. Since spoofing an identity is possible, the communicating parties may have no guarantees that the established communication is not being eavesdropped or tampered. Furthermore, in scenarios of pervasive computing and peer-to-peer networks, it is often not possible to rely on the existence of a globally trusted third party [48]. In these types of situations, ad hoc verification of keys via a secondary OOB channel is often the only solution.

The OOB channel provides robustness against attacks by introducing a second, independent communication channel. In order to eavesdrop or perform a man-in-the-middle attack (MITM) on the primary channel, the attacker is required to gain access also to the OOB channel during the pairing process.

2.1.2 Usability perspective

According to Kainda et al. [38], improving security often has a negative effect on usability. The authors state that the goal of security is to prevent or mitigate undesired actions while the goal of usability is to ease the process of the desirable actions. For example, the authors describe how some implementations of Digital Rights Management (DRM) in music industry have caused concern by restricting the genuine user to enjoying the purchased product only on a specific device. DRM may also restrict the genuine customer by allowing the product to be played only on specific platforms, e.g., a video game bought on Steam[1] can only run on authorized computers or an album bought on iTunes[2] can only be played on authorized devices.

Another example where security often hinders the user experience is the use of Completely Automated Public Turing test to tell Humans Apart (CAPTCHA). For instance, in order to login to a website, the user might be prompted with a challenge-response test, i.e., short puzzle, which can significantly increases the length of the login process. Based on a study conducted by Fidas et al. [19], CAPTCHAs are difficult for humans to solve. Even though the participants of the study had academic background and were familiar with CAPTCHAs, only 48.5% were able to solve the CAPTCHA on the first try.

The conflict between user experience and security is prominent in mobile applications that continuously prompt the user for permissions to specific resources, such as camera, microphone or pictures. This is cumbersome for the user as human attention is a limited resource [12] and the number of decisions expected from the user is high [45]. Continuous repetition of clicking through permissions may lead to the user accepting subsequent prompts without paying attention to the message [18], therefore granting unintended permissions to the application.

[1]Steam. https://store.steampowered.com/
[2]iTunes. https://www.apple.com/fi/itunes/

Secure systems have tendency of being broken by their users [37]. Security is determined by the weakest link and most often the user is the weakest link [12, 37]. These factors need to be taken into account when designing secure consumer products. Most often, the security of the product is not the main goal. According to Dhillon et al. [16], security and usability are always been an add-on to the product. Furthermore, the authors criticize that usability and security issues are not integrated into development process of systems, which results in systems *not aligned in terms of security and usability*.

While the implementations of OOB channels vary and are seen as a promising method for bootstrapping security in ad hoc networks, the benefits can be questioned with the added complexity of involving users and adding extra work for the user to complete the authentication process [38]. However, depending on the implementation, OOB channels can provide a viable alternative to many existing cumbersome and insecure mechanisms.

2.2 OOB channels in existing products and standards

OOB channels are widely used in commercial products and standards. Many modern smart home devices require the user to have a smartphone or a tablet as a companion device. This companion device might be used for both the initial setup process of the devices and for remote control of the devices once they are functional. In the next subsections, we will look at existing standards and commercial devices that rely on OOB channels.

2.2.1 Bluetooth

Bluetooth [11] is a widely deployed short range wireless communication standard. Bluetooth is used as the communication channel in wireless accessories, such as keyboards, computer mice, headphones, and speakers. It takes advantage of OOB channels in the pairing process between the devices before they can securely communicate.

The Bluetooth Secure Simple Pairing process consists of three steps [72]. In the first step, devices locate other Bluetooth devices and perform a Diffie-Hellman (DH) key exchange between the chosen devices. This is followed by an authentication step, which is performed over an OOB channel. The

Bluetooth standard supports three OOB authentication methods that verify the performed key exchange [11, 72]:

1. **Numeric Comparison**. In this method, the user is required to compare and confirm that the short six-digit codes are identical on the displays of the two devices. In addition to verifying the key exchange, it provides confirmation that the user has paired correct devices. This pairing method is suitable for pairing devices with displays and some input capabilities for the user to confirm the pairing.

2. **Passkey Entry.** In this method, the first device displays a six-digit passkey, which the user is required to enter into the second device. This pairing method is suitable for scenarios where the other device only has input capabilities but lack a display, e.g., a keyboard.

3. **Out of Band**. This method relies on a high-bandwidth two-directional OOB channel, e.g., Near-Field Communication (NFC). The channel is utilized for discovering other devices and exchanging or transferring cryptographic information needed for the device pairing process.

Finally, in the third step, both devices confirm the DH key exchange by exchanging message authentication codes (MAC) computed from the key exchange parameters [72].

If either of the devices has very limited input and output, Bluetooth supports unauthenticated Diffie-Hellman key exchange. This **Just Works** method performs the key exchange and follows the Numeric Comparison method without requiring the user to confirm the matching checksums [11, 72]. Compared to the authenticated methods, it only protects against passive eavesdropping while remaining vulnerable to man-in-the-middle attacks.

2.2.2 Nest

Nest Labs[3] is a smart home appliances manufacturer. These smart appliances range from cameras, thermostats, doorbells, locks to various alarms. Nest devices require Internet access, the Nest application on a smart phone or a tablet and a Nest account. In most products, the initial pairing between the devices and the application requires Bluetooth [52]. While the installation and pairing process may vary slightly, the process follows a similar pattern

[3]Nest Labs. https://nest.com

for each product. After choosing the physical location for the device, the device requires access to the Internet. This requires the user to select the correct Wi-Fi network and enter the password into the device.

After connecting the device into the local network, the typical pairing process proceeds by pairing the device to the Nest application. In older products, this is done via an OOB channel in the form of PIN code displayed by the device. For instance, when pairing a Nest thermostat to an application, the thermostat displays a seven-digit PIN code which the user is required to type into the application. However, newer products must be paired before connecting to the Wi-Fi network. Instead of a PIN code, these products are able to acquire the pairing information from the static QR codes, which are printed on the device or on a separate paper within the retail packaging. After pairing the device to the application, the device requires access to Wi-Fi network. Wi-Fi credentials are delivered to it via the Nest application.

For authorizing third-party products to access the Nest ecosystem, Nest employs OAuth 2.0 authorization URLs and PIN codes [54]. In order to authorize third-party party products into the Nest ecosystem, the user is first required to create an account on the third-party party developer site or mobile application [53]. Thereafter, the user needs to enable Works With Nest connection option from the third-party party developer site or the application. This initiates the OAuth 2.0 authorization process, in which the Nest API is requested for an access token. The request is in the form of a URL or a PIN code. The Nest API supports Time-To-Live (TTL) values of 10 minutes for URLs and 48 hours for PIN codes [51]. URLs are the preferred method for authenticating a device into the system; however, if the device is not capable of displaying web content, PIN codes are a supported method as well [54]. In PIN code authorization, the user types in the PIN code displayed by the Nest application into the third-party party device to request an access token. After the access token request has been made, the user needs to grant permission from the Nest application. This allows the Nest API to respond to the request with an access token. Access token grants the authorized third-party party device a varying degrees of access to the Nest API. This token based authorization allows the user to revoke device's access at any time.

2.2.3 Chromecast

Google Chromecast[4] is a digital media player which allows the user to cast digital content to the television from a smartphone or tablet. The device is attached to the television with HDMI and takes commands via a local wireless network. However, it relies on the Google Cast application for device management and configuration.

In order to pair the Chromecast to the application, the user needs to select the correct wireless network created by the Chromecast device. This wireless network information, i.e., Service Set ID (SSID), is displayed by the television. After connecting to the network, the application displays an OOB verification message in the form of four digits. This step requires the user to compare and confirm that the codes are the same both in the application and on the television. After the verification step, the application requires the user to select the wireless home network and enter a password in order for Chromecast to connect to local Wi-Fi and the Internet.

Another OOB method in newer Chromecast devices is ultrasound, which is used in guest mode[5] for pairing Chromecast with a guest device. The Chromecast device allows a connection with a guest device without the guest having access to the wireless home network. Instead of the guest user having to manually enter a PIN code displayed on the television, the Chromecast device sends the PIN code via speakers using high-frequency sound. This is called an ultrasonic token, which is picked up by the guest user's smartphone. After the pairing process, the guest user can then cast digital media to the device. However, the PIN code is valid for 24 hours at a time.

2.2.4 Apple

Apple manufactures various wireless accessories, such as keyboards and headphones for their smartphones, tablets and computers. Similarly to other wireless devices, they require pairing before they can securely communicate. Most Apple devices communicate with Bluetooth and follow the pairing process defined by the Bluetooth specification. Devices with limited input, such as AirPods[6] headphones, rely on the just-works pairing method. The initial pairing process is triggered by opening the charging case of the headphones

[4]Chromecast. https://support.google.com/chromecast/
[5]Chromecast Guest mode. https://developers.google.com/cast/docs/guest_mode
[6]AirPods. https://www.apple.com/airpods/

near another Apple device, e.g., iPhone smartphone. User is then required to accept the UI prompt displayed on the smartphone, which finalizes the pairing process.

For pairing Apple devices with displays, Apple employs a custom moving image [5]. However, this method requires the user to already have one Apple product with a working camera to interpret the image. An example of this would be the pairing process between an Apple Watch[7] and an iPhone. After turning on the watch, the nearby iPhones with Bluetooth connection enabled are prompted with a UI notification for pairing them with the watch. Accepting the prompt opens a camera on the phone, and the watch begins to display a swirling image which the user is required to capture. The user is required to focus the image to the iPhone camera. The pairing is completed after successfully capturing the image. However, for scenarios where iPhone camera is not available, Apple Watch supports manual pairing. In this method, the user identifies the watch on the phone with a five digit identifier shown on the watch. After choosing the watch from the available Bluetooth devices list, the watch displays a six-digit code which the user is required to input to the phone. This approach follows the passkey entry method of Bluetooth Secure Simple Pairing.

Apple Home[8] is an application that allows multiple smart home devices to be controlled with Apple devices. These smart home devices require an Apple device, such as an iPhone or an iPad in their setup process and configuration [4]. The application allows automation and remote control. In order to add devices to the Home application, users are required to scan QR codes or NFC tags on compatible devices.

While the pairing process varies per device, as most devices it is fairly similar to Nest. For example, a third-party party smart device, FIBARO sensor[9], follows the typical pairing process pattern. After it is installed to a physical location, the device is powered up by removing a battery blocker. It will emit a blue glow indicating that it is ready to be paired. The device is found with Bluetooth and requires scanning a QR code or manually entering an eight-digit PIN to complete the OOB pairing process. The QR codes are static and are either printed on the device or found inside the retail package.

[7]Apple Watch. https://www.apple.com/watch/
[8]Apple Home. https://www.apple.com/ios/home/
[9]FIBARO Sensor. https://manuals.fibaro.com/hk-door-window-sensor/

2.2.5 Group messaging

Modern popular group messaging applications, such as Telegram[10], WhatsApp[11] and Signal[12], support end-to-end encryption with OOB verification. The verification requires users to compare information shown on each other's devices.

In Telegram, encrypted communication is established with a Diffie-Hellman key exchange [74]. Based on the key exchange, a picture is generated with additional textual representation of the keys [75]. To confirm the end-to-end connection, users compare the pictures. If the images are identical on all the participating devices, the users have strong guarantee that the connection is secure.

In order to confirm encrypted end-to-end connection in WhatsApp and Signal, users either scan a QR code or visually compare a 60-digit number [80]. The 60-digit number is concatenated and hashed combination of each user's keys and thus unique to each conversation. To take advantage of the QR codes, the user scans the QR code shown on the other user's device. After scanning the QR code, WhatsApp displays a green check mark if the QR scan yields an identical result to the one captured on the verifying smartphone.

A study conducted by Naor et al. [50] showed that the approach of numerical comparison in WhatsApp remains vulnerable to man-in-the-middle attacks when the user is *lazy* and only compares half (or less) of the 60-digit OOB value.

2.2.6 One time passwords

One Time Password (OTP) sent via the Short Message Service (SMS) is one of the most used multi-factor authentication and authorization schemes [63]. In order for the user to authenticate into a system, the user needs to show that they know the user ID and password and have the access to a registered mobile phone. SMS is adopted by many service providers from banks to online stores and social networks [69]. However, in some cases the way these methods are implemented does not protect against phishing and man-in-the-middle (MITM) attacks, e.g., in scenarios where the login screen is fake.

[10]Telegram. https://telegram.org, and Secret chat. https://telegram.org/faq
[11]WhatsApp. https://whatsapp.com
[12]Signal. https://signal.org

Some service providers, including various internet forums, use email to verify users with proper email address, however, Grassi et al. [30] suggests that email should no longer be considered a secure and valid option as an out-of-band authentication channel.

Most banking systems in Finland take advantage of key lists in online banking and authentication. These are small paper sheets with printed random numbers, which are asked during login process and during confirmation of payments and wire transfers. Furthermore, these numbers are used for verifying the identity of the user when login into public services (TUPAS). However, from September 2019 TUPAS is no longer considered to fulfil the requirements for strong authentication [77], and most banks now provide an application to generate OTPs.

Chapter 3

EAP-NOOB

In this chapter we provide a high level overview of the Nimble Out-of-Band for EAP (EAP-NOOB) protocol [7]. Extensible Authentication Protocol (EAP) is an authentication framework with support for multiple methods of authentication [79]. Currently, there are numerous different EAP methods. Some of the most commonly deployed EAP authentication types include EAP-TLS [70], EAP-PEAP [59], and EAP-TTLS [25].

EAP is typically used for wireless network access authentication in enterprise environments. EAP is often implemented over data link layers, e.g., IEEE 802 or Point-to-Point protocol (PPP) [33]. EAP can be implemented on dedicated links and switched circuits, in addition to wired and wireless links.

3.1 Nimble out-of-band (NOOB)

Nimble out-of-band authentication for EAP [7] is a new EAP method. It is an open standard and a generic protocol intended specifically for bootstrapping varying Internet-of-Things (IoT) devices. As is the case with many IoT devices, these devices may have limited input and output capabilities.

EAP-NOOB does not require the devices to have any pre-configured authentication credentials. Instead, device configuration and registration to a server database along with ownership information and authentication of newly created credentials are all performed during the initial device deployment. In this regard, EAP-NOOB is unique compared to most other EAP methods that assume some credentials have been provisioned.

Since EAP-NOOB does not require any pre-configured credentials it follows the common device pairing approach of Diffie-Hellman (DH) key exchange over insecure network. However, the key-exchange alone does not provide authentication and, therefore, the key-exchange is authenticated with a message sent over an OOB channel. This prevents impersonation and man-in-the-middle attacks on the in-band channel.

In addition to the DH key-exchange, EAP-NOOB contains two security features used in the OOB message. The protocol relies on the secret nonce (Noob), which is used as the first authentication feature. The secret nonce is utilized for mutually authenticating the session key. The second authentication feature is the cryptographic fingerprint (Hoob), which is used to verify the integrity of the key exchange. The end point that receives the OOB message may use the fingerprint to expose impersonation and man-in-the-middle attacks on the in-band channel. EAP-NOOB follows the EAP-TLS Authentication Protocol [70] terms defined for the authenticator, the peer and the server [7]. The entity initiating the EAP authentication is referred to as the authenticator while the peer is described as the entity responding to the authenticator. The entity terminating the EAP authentication is referred to as the server.

Another feature of EAP-NOOB protocol is that it allows peer devices to scan for possible networks and, based on the results, generate multiple dynamic OOB messages. These messages are relatively long and require partly automated means of transfer. The main purpose of an OOB message is to mutually authenticate the peer devices and the server. These messages can be sent from the peer to the server (shown in Figure 3.2) or from the server to the peer (shown in Figure 3.1). For instance, a peer devices, such as a smart display, may show OOB message encoded in QR code, while a smart speaker may use NFC module or sound to transmit an OOB message.

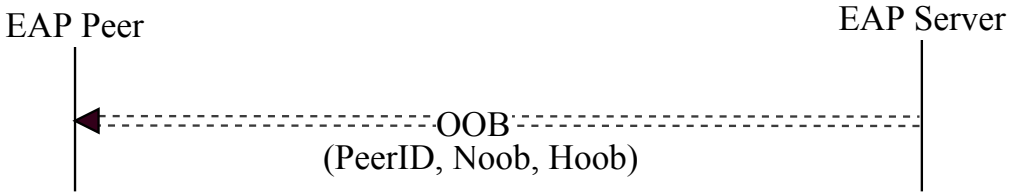


Figure 3.1: OOB step, in which OOB message originates from EAP server and is delivered to peer via user-assisted OOB channel [7].

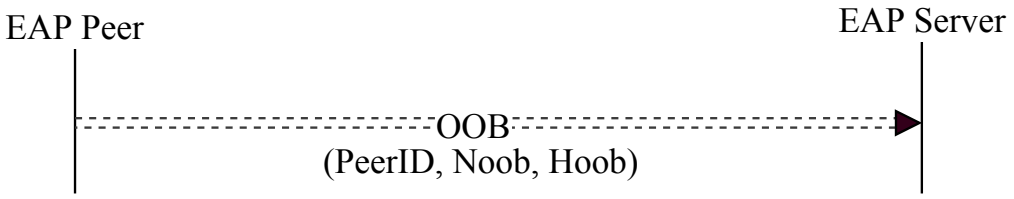


Figure 3.2: OOB step where OOB message originates from peer and is delivered to EAP server via user-assisted OOB channel [7].

OOB messages have limited time to be delivered from the peer to the server or vice versa. The protocol [7] suggests a fairly generous timeout value of 3600 seconds. During this time, the OOB message is valid and should be delivered. However, the devices generate new nonces (Noob) to guarantee freshness. The recommended refresh cycle is half of the timeout value of the OOB message. If the message is not delivered within the time limit, it expires. Delivering an expired OOB message results in the sender not recognizing the OOB message during key confirmation phase.

The protocol specification suggests that it may be convenient to encode the OOB message as a URL [7]. This method is suitable for scenarios where the OOB message is sent from the peer to the server (Figure 3.1). The OOB message is delivered to the authentication server by visiting the URL. The URL consists of server domain name and additionally carries the PeerId, secret nonce (Noob) and fingerprint (Hoob) as query string parameters.

The EAP-NOOB protocol specification sets some restrictions and suggestions for the URL. The server domain name has a maximum length of 60 characters. The PeerId is provided by the server with a maximum length of 60 bytes and should not include the '+' sign. To shorten the query parameters, the specification suggests the PeerId length of 22 characters resulting from base64url encoding. The secret nonce (Noob) and the fingerprint (Hoob) are both specified as 16-byte values, which are encoded into character strings with base64url encoding. After encoding the length of both strings is 22 characters. This results in approximately a 70-130 character string that needs to be transferred over an OOB channel. However, the length may vary depending on the varying length of the used data fields, i.e., server name or PeerId.

When the EAP-NOOB protocol relies on user assisted OOB channel with OOB messages encoded as URLs, the Internet connection (e.g., 3G, 4G or

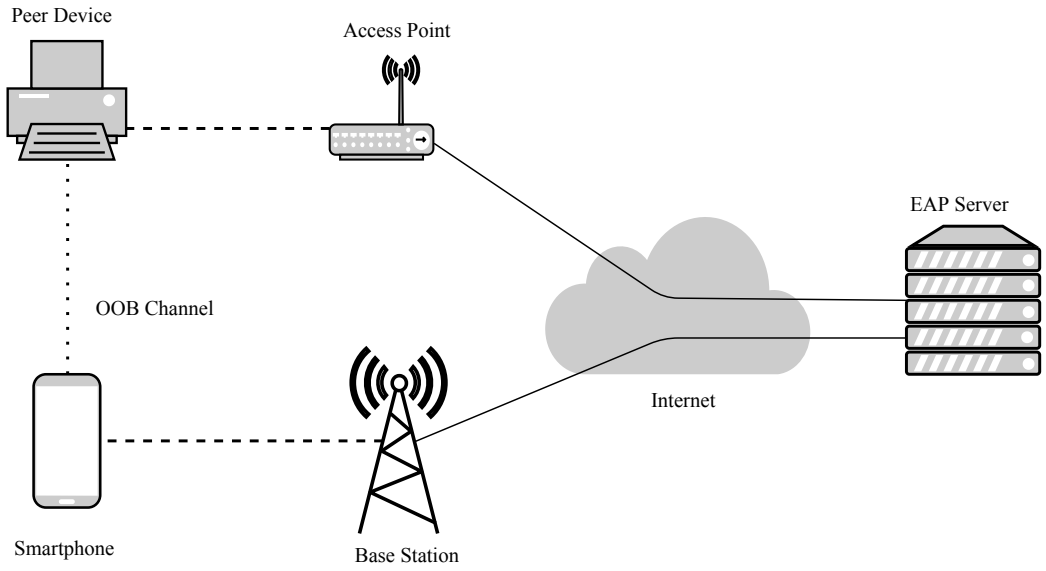


Figure 3.3: EAP-NOOB setup

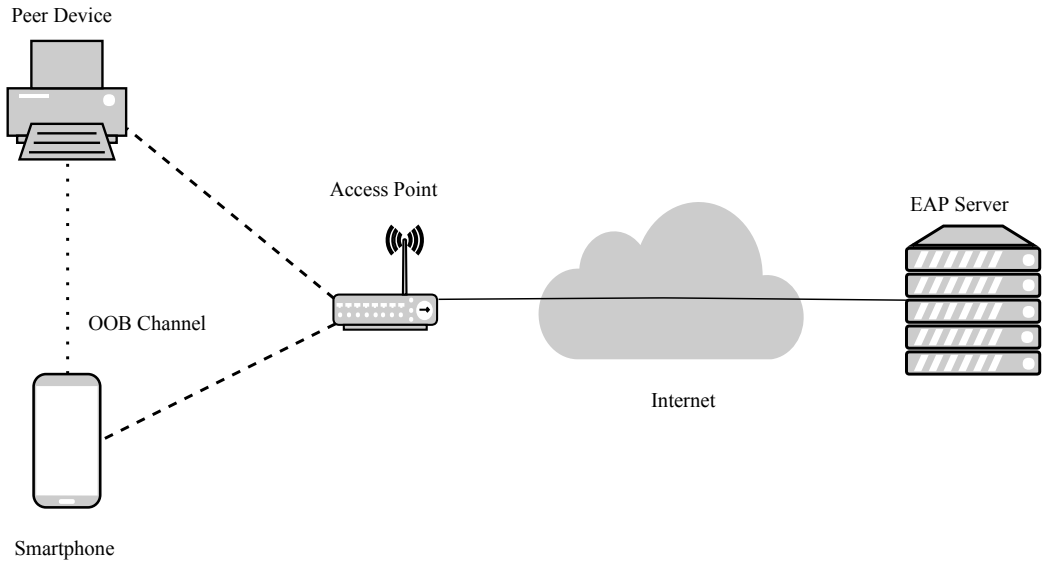


Figure 3.4: EAP-NOOB setup where peer device and user equipment share the same network, exposing the OOB message to phishing attacks.

5G) of the user's smartphone is utilized for delivering the OOB message. This scenario is shown in Figure 3.3. However, scenarios where the peer device and the user smartphone connect to the same wireless network may occur (shown in Figure 3.4). If the access point (AP) is compromised, the user might be exposed to phishing attacks if the user does not check if the URL is HTTPS or if the server name is not correct. While the EAP-NOOB protocol does not specify OOB channel type, it supports simple authentication via URL. These URLs can be encoded in QR codes or NDEF tags and processed with a smartphone. This is ideal for IoT devices with limited input or output. For instance a simple printer may output a printed QR code on paper, or it may have an NFC module that can be scanned with a smart phone. Another example could be a public information panel that during the setup process can display the OOB message encoded as a QR code. Both scenarios rely on the user for scanning the OOB message and finalizing the registration process by visiting the URL. In this thesis, we focus on scenarios where the OOB message is sent from the peer to the server.

Table 3.1: OOB message format as URL.

Data field	prefix	Example data
ServerURL	https://	https://example.com/Noob
PeerId	?P	?P=ZrD7qkczNoHGbGcN2bN0
Nonce	&N	&N=rMinS0-F4EfCU8D9ljxXA
Fingerprint	&H	&H=QvnMp4UGxuQVFaXPW_14UW

Example of data fields in OOB message when encoded as URL are shown in Table 3.1. ?P= indicates the PeerId which the server has allocated for the peer device. &N= indicates the secret nonce (Noob) and &H= indicates the cryptographic fingerprint (Hoob).

The EAP-NOOB specification [7] has seen multiple iterations. Experimental implementations have been created to examine the protocol [49, 76]. Furthermore, the EAP-NOOB protocol has been modeled with mCRL2[1] formal modeling language [61] to simulate protocol behavior and with ProVerif[2] tool for verifying its security characteristics [67].

[1]mCRL2. https://www.mcrl2.org/web/user_manual/index.html
[2]ProVerif. https://prosecco.gforge.inria.fr/personal/bblanche/proverif/

Chapter 4

Out-Of-Band Channels

There are numerous ways of implementing OOB channels. This chapter describes the technologies behind NFC, QR codes and sound for implementing the three different OOB channels, which are used for transmitting an OOB message. In addition, we describe the default device behavior when interacting with QR codes and NFC. Furthermore, we examine the advantages and disadvantages of each technology as an OOB channel as well as describe the known vulnerabilities of each approach.

4.1 NFC

Near-field communication (NFC) is a standards-based wireless communication technology [56]. It utilizes high-frequency magnetic alternating fields for transmitting data between two devices. The protocol operates in the frequency of 13.56 MHz. Due to the transmission taking place only inside the generated magnetic field, the typical transmission range is limited to 20 centimeters [21]. In order to initiate communication between two NFC devices, the devices are assigned as the NFC initiator and the NFC target. The communication between the two NFC devices is always initiated by the NFC initiator, and the NFC target responds. NFC specifies an active mode and a passive mode for device interaction. These provide three varying modes of operation:

1. Active mode - Peer-to-peer. This operation mode is employed for peer-to-peer communication between NFC devices (shown in Figure 4.1).

2. Passive mode - Reader emulation. This is used for interaction with Radio-frequency identification (RFID) transponders, i.e., the device is able to act as a RFID reader (shown in Figure. 4.2).

3. Passive mode - Card emulation. The card emulation is used for device to act as a RFID transponder, i.e., the device is able to emulate RFID card behavior (shown in Figure 4.3).

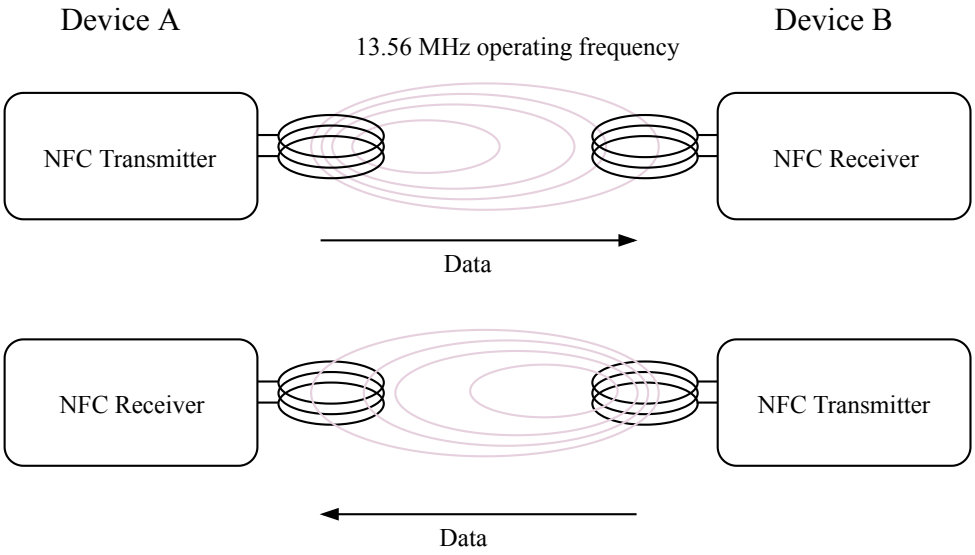


Figure 4.1: Peer-to-peer mode [21]

In the active mode, the NFC initiator generates a magnetic field for sending the initiator signal. If this field is perceived by the peer device, it takes the role of an NFC target. The communication is based on Amplitude-Shift Keying (ASK) modulation. For back and forth communication, NFC devices take turns in generating the alternating magnetic fields as shown in Figure 4.1. Thus, both devices requiring a power source. Data is always sent from transmitter to receiver [21].

The passive mode is similar to the active mode and can be divided into reader emulation and card emulation modes. However, compared to the active mode the NFC initiator will not stop generating the magnetic field when it stops transmitting data. This allows the NFC target to transmit data to the NFC initiator via load modulation. Consequently, this method relies on the NFC initiator to generate the magnetic field. The approach is

similar to many RFID systems where the target device is similarly powered by the initiator device. Therefore, an NFC Initiator may act as a RFID reader as illustrated in Figure 4.2. This enables the device to communicate

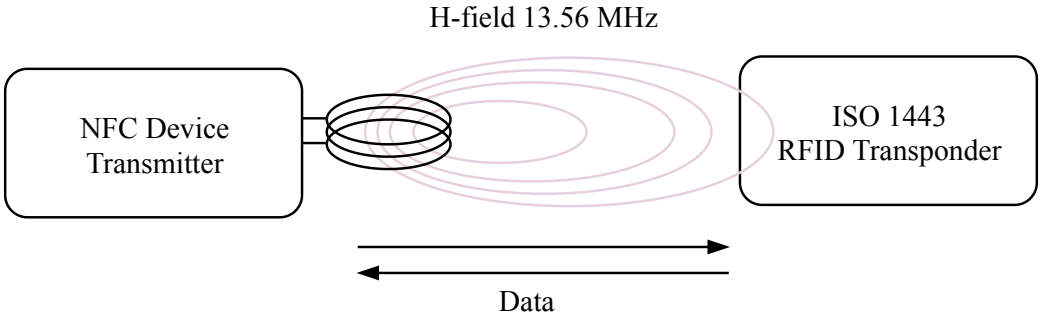


Figure 4.2: Passive Mode - Reader Emulation [21]

with various transponders, such as contactless smart cards and tags based on the ISO/IEC 14443 standard [21].

The passive mode is very power efficient for the target device. It is a viable option for systems with low power requirements due to power being supplied by the initiator device. This method has numerous advantages and options for practical implementations [21]. This can be seen in the wide use of smart cards, such travel cards and tags.

Finally, the card emulation mode, shown in Figure 4.3, enables the NFC device to act as an NFC target, i.e., NFC tag. The NFC device transmits data with load modulation allowing RFID readers based on the ISO/IEC 14443 standard to communicate with the device. For the RFID reader, the NFC device appears as a contactless smart card.

NFC Forum lists four types of NFC tags with special characteristics.

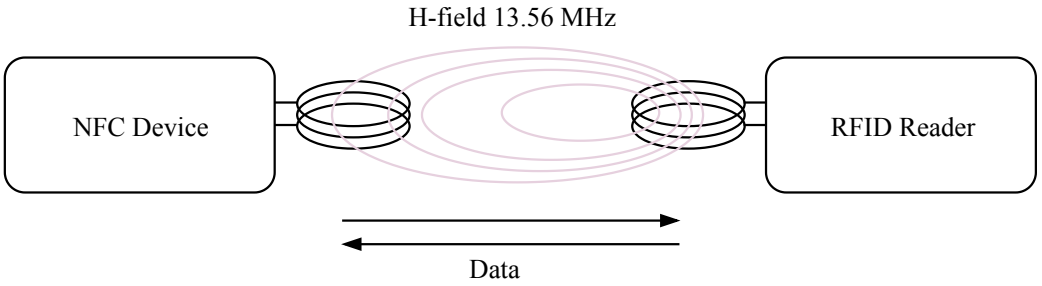


Figure 4.3: Passive Mode - Card Emulation [21]

Types I, II and IV are based on the ISO-14443 standard, while Type III is based on the ISO-18092 standard [32]. The Type V is not specified by the NFC Forum. It is a proprietary tag defined by NXP Semiconductor. Nevertheless, it is the most common type of NFC smart card [32]. The characteristics of each tag type are below:

- Type I — read only or read-write, memory size of 96 B to 2 kB, transfer speed of 106 kbps, no data collision protection.
- Type II — read only or read-write, memory size of 96B to 2kB, transfer speed of 106 kbps, anti-collision support.
- Type III — read only or read-write, memory size of up to 1MB, transfer speeds of 212 or 424 kbps, anti-collision support.
- Type IV — read only or read-write, memory size of 2, 4 or 8 kB, transfer speeds of 106, 212 or 424 kbps, anti-collision support.
- Type V — read-write, memory size of 192, 768 or 3584 B, transfer speed 106 kbps, anti-collision support.

The data exchanged between NFC devices and tags is formatted in the NFC Data Exchange Format (NDEF). Regardless of the technology or tag type, all NFC devices and tags support NDEF [32]. A single NDEF Message is constructed with one or more NDEF Records. An NDEF Record consists of a record header and a data payload. The record header is metadata, which is used to interpret the payload. The record header includes data fields, such as Type Name Format (TNF) and Message Flags field, Type Length, Payload Length, ID Length, Payload Type and Payload ID [32]. The NDEF message structure is illustrated in Figure 4.4

The TNF describes how to interpret the data; i.e., it specifies the format of the payload. Currently, there exists seven defined values for the TNF field [55]:

1. Empty
   The record does not have a type or payload. TNF = 0x00.
2. Well-Known
   The type follows the Record Type Definition (RTD) name format specified by NFC Forum. TNF = 0x01.

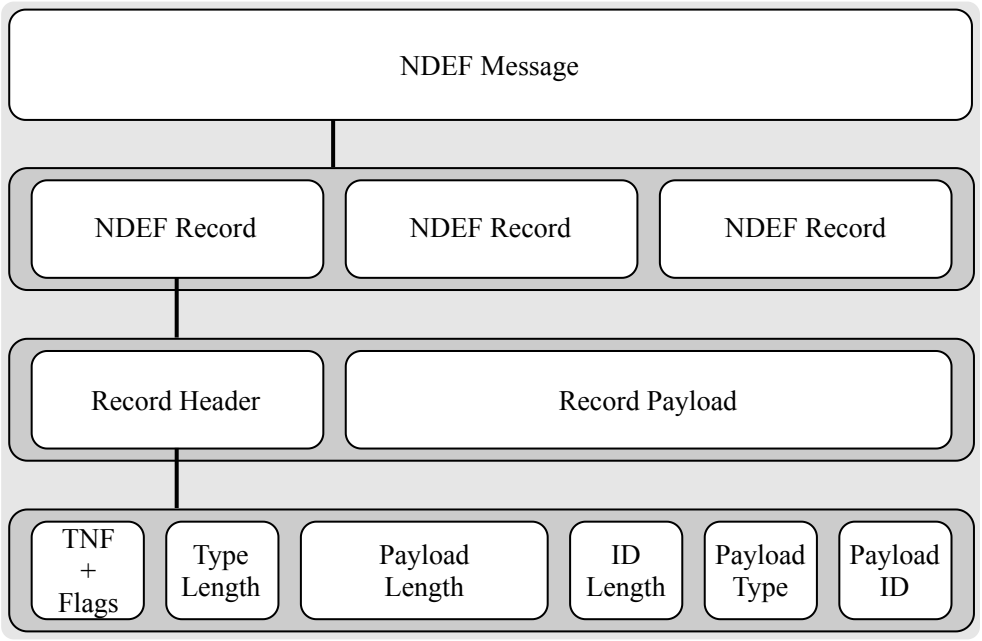


Figure 4.4: NDEF message structure [32]

3. Media-Type
   The type follows the media-type Multipurpose Internet Mail Extensions (MIME) as specified in RFC 2046 [24]. TNF = 0x02.

4. Absolute URI
   Uniform Resource Identifier (URI) type as specified in RFC 3986 [10]. TNF = 0x03.

5. External
   External type as specified by NFC Forum RTD specification. This can be utilized for user defined values, such as the Android Application Record for opening applications. TNF = 0x04.

6. Unknown
   Unknown type with type length zero. This is utilized for NDEF payloads without pre-defined processing. TNF = 0x05.

7. Unchanged
   Chunked payload included in the middle and last record. This TNF value also requires the type length to be zero. TNF = 0x06.

8. Reserved
   Reserved for possible later use by the NFC Forum. TNF = 0x07.

While the size of the NDEF record payload is limited to $2^{32} - 1$ bytes, the size of the NDEF message is not restricted [32]. Furthermore, there is no limitation on how many NDEF records one NDEF message can hold. However, in most use cases the size of the NDEF message is limited by the usability and convenience, as well as the hardware capabilities of the tag, e.g., computational capacity or storage size. In smart tags (listed earlier), the size limits are defined by the tag type.

4.1.1 Default Android NFC reader

Most if not all Android devices with NFC modules support NFC communication. When new NFC tags are discovered, they are parsed and analyzed by the Android tag dispatch system [28]. After parsing and analyzing the NFC tag, the tag dispatch system attempts to initiate the correct application interested in the tag data. This is performed by reading the first NDEF formatted data record and looking for a MIME type or an identifying URI. If the search is successful, the data is encapsulated inside the intent ACTION_NDEF_DISCOVERED consisting of a 3-bit Type Name Field (TNF), variable length type and payload. However, if the search fails and the NDEF record does not contain MIME type or identifying URI, the tag information and payload are encapsulated in the ACTION_TECH_DISCOVERED intent. Furthermore, if no activity picks up on ACTION_NDEF_DISCOVERED or ACTION_TECH_DISCOVERED, then the ACTION_TAG_DISCOVERED intent is started. Illustration of the Android tag dispatch system is in Figure 4.5.

If intent is tied to an activity, Android tries to launch an application with the intent. If there exists multiple applications that qualify for processing the intent, Android prompts the user with a list of the applications which may process the intent (Figure 4.6 (a)). Applications declare an intent filter, which provides the Android tag dispatch system with the data types they can process. No action is taken if none of the capable applications are interested in the tag data.

In addition, Android searches the complete NDEF message for Android Application Records (AAR) [28]. AARs are used for either launching an installed application responding to the package name inside the AAR or for

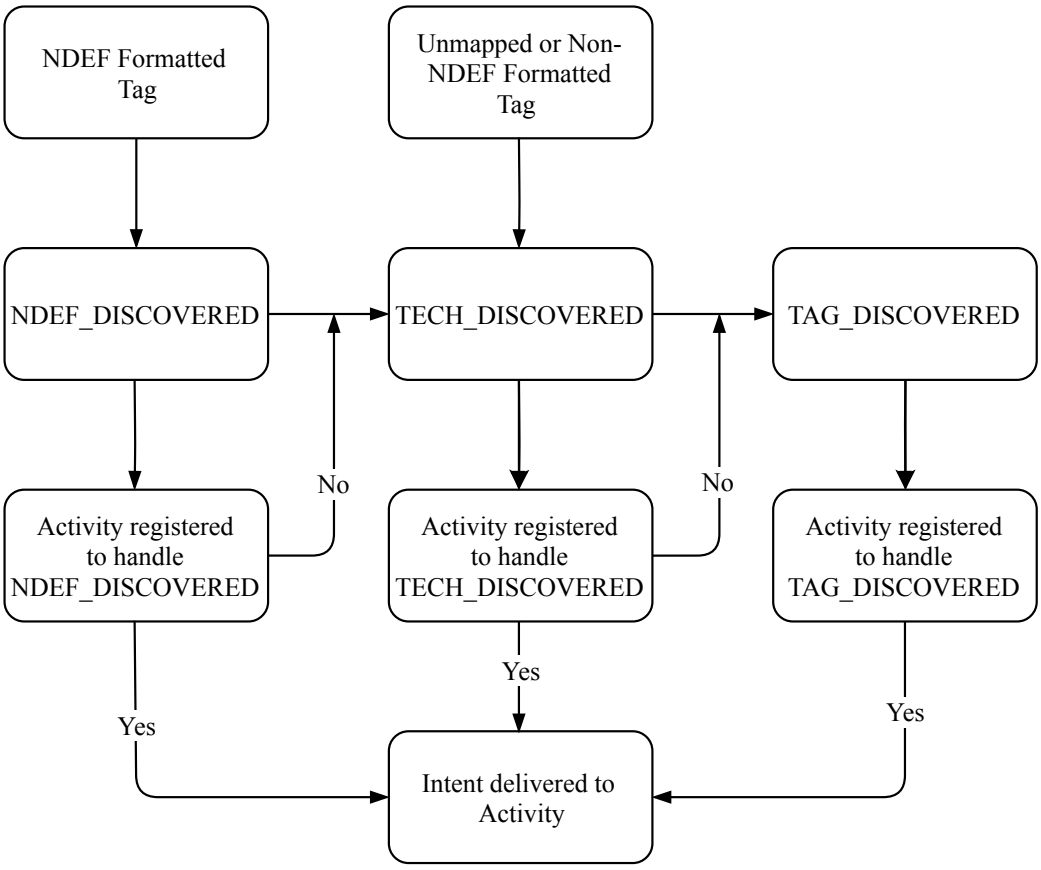
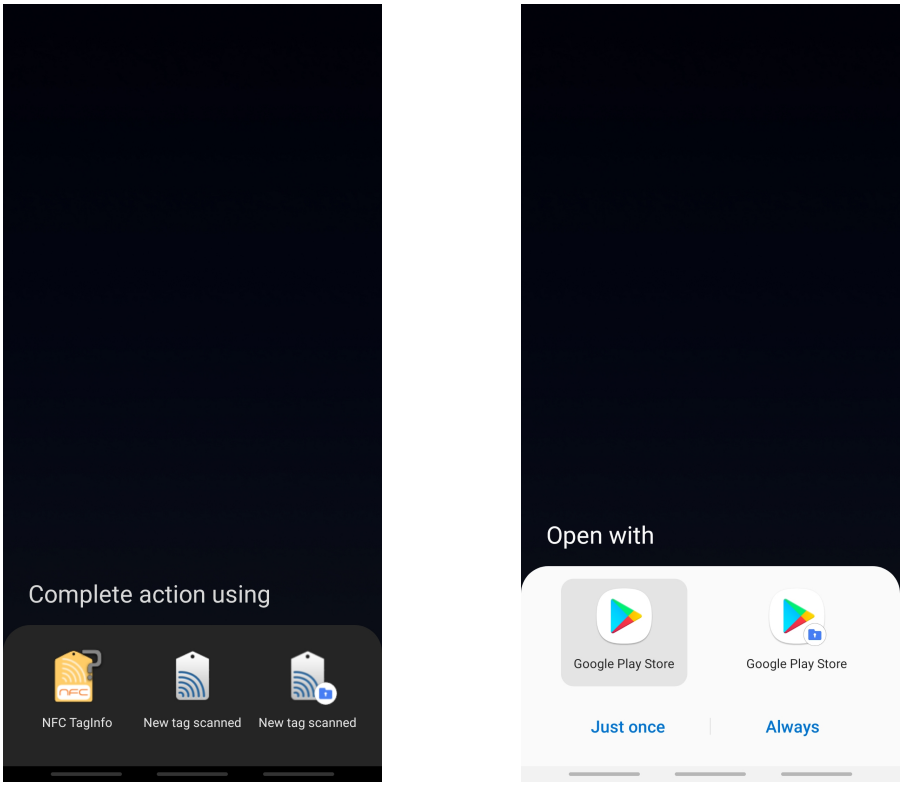


Figure 4.5: Illustrating Android's tag dispatch system [28].

launching Google Play Store search for the package name. AARs provide strong certainty that the designated application is launched when scanned [28]. However, this automated behavior can be circumvented by utilizing third-party readers that do not automatically execute AARs.

Typically, Android smartphones with NFC hardware scan for available NFC tags when NFC is enabled in the device settings and the screen is unlocked. Scanning for tags with Android device is demonstrated in Figure 4.6, where (a) multiple applications may process tag data, and (b) AAR contains an application package name not installed on phone, which prompts

Google Play Store.



(a) Android prompting user action for processing tag data

(b) Android prompting Google Play Store for application

Figure 4.6: Screen capture of Android tag dispatch system.

The first NDEF record is analyzed by the system to determine which activity is followed. This is done without any user action. According to Google [28], it is purely a design decision. The reasoning behind the decision is that the required manual user input would possibly break the NFC connection because of the movement caused by user.

If the first NDEF record is a URL, Android opens the default Internet browser and the rest of the NDEF message is not displayed. However, if there are multiple Internet browsers, then the user is prompted to choose one of them to open the URL. There are slight variations in the process between smartphone manufacturers. For example, in order for the default NFC reader to display all of the NDEF records in Samsung Galaxy S8, the first record must be formatted as plain text to avoid the Internet browser's intent filter.

4.1.2 Vulnerabilities in NFC

NFC has been studied extensively and some vulnerabilities have been identified. NFC is vulnerable to a number of attacks that can be categorized as follows [15, 31]:

1. **Eavesdropping.** As a wireless communication method, NFC is vulnerable to eavesdropping. For intercepting NFC communication between two devices, the attacker must have access to the location where the communication takes place. Furthermore, taking advantage of larger and more powerful antennas than currently found in smart phones, the attacker can eavesdrop radio emissions from the communication from even larger distances [15, 31]. Haselsteiner & Breitfuß [31], list various factors that determine how close an attacker is required to be in order to successfully eavesdrop the NFC communication. One of these factors is Radio-Frequency (RF) characteristics of the sender device, such as geometry of the antennas, shielding of the device and power of the NFC transmitter. Another factor regards the characteristics of the attacker device, which include receiver quality, RF signal decoder quality, and geometry of the antennas. Furthermore, the environment and location is a factor, e.g., background noise or interfering walls. According to Haselsteiner & Breitfuß [31], NFC active mode is vulnerable to eavesdropping up to 10 m distance while passive mode is vulnerable to 1 m distance.

2. **Data Corruption.** Data corruption is a form of Denial of Service (DoS) attack, in which the attacker is able to corrupt the data between two NFC interfaces. This is achieved by sending a more powerful signal during the NFC exchange. Furthermore, the malicious transmission may halt the communication between the sender and receiver. Therefore, the effects of data corruption attack are temporary for communication between two devices. However, data transmitted to the NFC tags remains corrupted.

3. **Data Modification.** Data Modification attack refers to a method where the attacker is able to tamper the actual data. According to Chattha [15], vulnerability to data modification attack can be eliminated by utilizing active mode. This is because the attacker would have to generate the RF signal that perfectly overlaps with the original

signal to prevent the signal from reaching legitimate receiver's antenna [31]. In addition, NFC devices are able to monitor the RF field before initiating communication.

4. **Data Insertion.** A system is vulnerable to data insertion attack if the NFC target, i.e., device responding to the message, is delayed. This allows the attacker to insert tampered data into the communication. However, the data may be corrupted if the delayed device begins the response at the same time.

5. **Man-in-the-middle.** This attack follows the classical method of the attacker inserting itself between two parties without alerting them. In this method the attacker is able to monitor and capture all the data between the exchange while routing the data between the two legitimate parties.

In addition to these, NFC is vulnerable to **relay attacks**, where attacker implements proxy devices to relay the communication which extends the range of the NFC communication [23].

4.2 QR Code

Quick Response (QR) code is a two-dimensional bar code. It was standardized in 2000 as a ISO/IEC18004 [35, 71]. QR codes provide a method for displaying data in machine readable form. Due to QR codes having the capacity to store more data than bar codes, QR codes have become more popular and an appealing choice for encoding data [40]. Furthermore, QR codes can be encoded approximately in one tenth of the size of a one-dimensional barcode with the same quantity of information.

Typical bar code is limited to 20 numeric characters [40]. Conversely, QR codes can contain up to 7089 numeric characters [35]. For storing alphanumeric data the size is limited to 4296 characters. The volume of data a QR code can hold is displayed in Table 4.1. However, the more data is stored inside a QR code the more complex the image becomes. Complex QR codes take longer to read and might require better hardware, such as optics and camera sensor. Utilizing data compression algorithms a QR code has been encoded to contain more than 4 Mb of data [78].

Table 4.1: QR code data types and maximum number of characters.

Data Type	Maximum characters
Numeric	7089
Alphanumeric	4296
Binary (8 bit)	2953
Kanji	1817

QR code structure is shown in Figure 4.7. QR code has a clear identifiable feature of three square position detection patterns and a number of square alignment patterns. For decoding the QR code, the quiet zone is first identified while alignment patterns are used for correcting the distortion of the QR code [78].

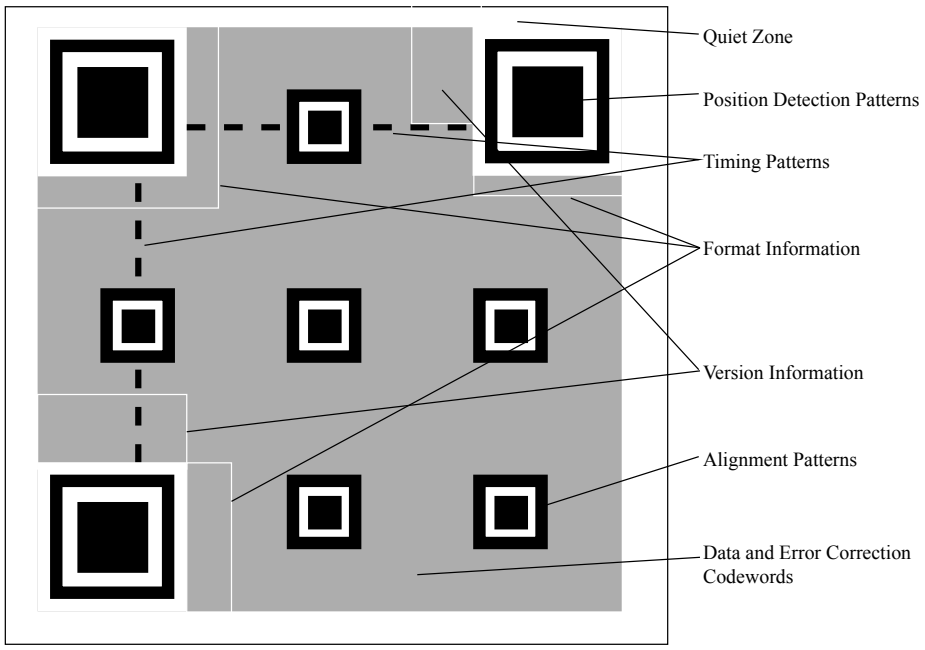


Figure 4.7: QR code structure [34]

According to Liu et al. [46], QR code is not always reliable due to varying conditions. These are bad lighting, skewing, low contradistinction, highlight spots and other mixed environmental conditions. To combat against scenarios where the code is partially destroyed or covered, QR code supports four levels of error correction and masking [35]. These error correction levels are L (7%), M (15 %), Q (25 %), and H (30 %) where the percentage equals

the approximate recovery capacity. However, higher error correction levels increase the area reserved for error correction while decreasing the area for actual data. An additional feature that increases the QR code readability is masking. It balances the dark and light modules within the QR code.

QR codes support multiple URI schemes. These are encoded in text format within the QR code following the intended URI format. These allow quick access to websites, locations, phone numbers, and Wi-Fi configuration. Some of URI scheme examples are displayed in Table 4.2.

Table 4.2: QR code URI schemes with example data.

Data Type	Action	Example data
URL	Open URL	https://aalto.fi/
Location	Open location	geo:60.186826,24.822062
Message	Compose a message	sms:number
Message	Send a pre-written message	SMSTO:number:message
Email	Compose an email	mailto:some.one@email.org
Phone number	Call a phone number	tel:number
Wi-Fi	Join a Wi-Fi network	WIFI:S:SSID;P:password;;

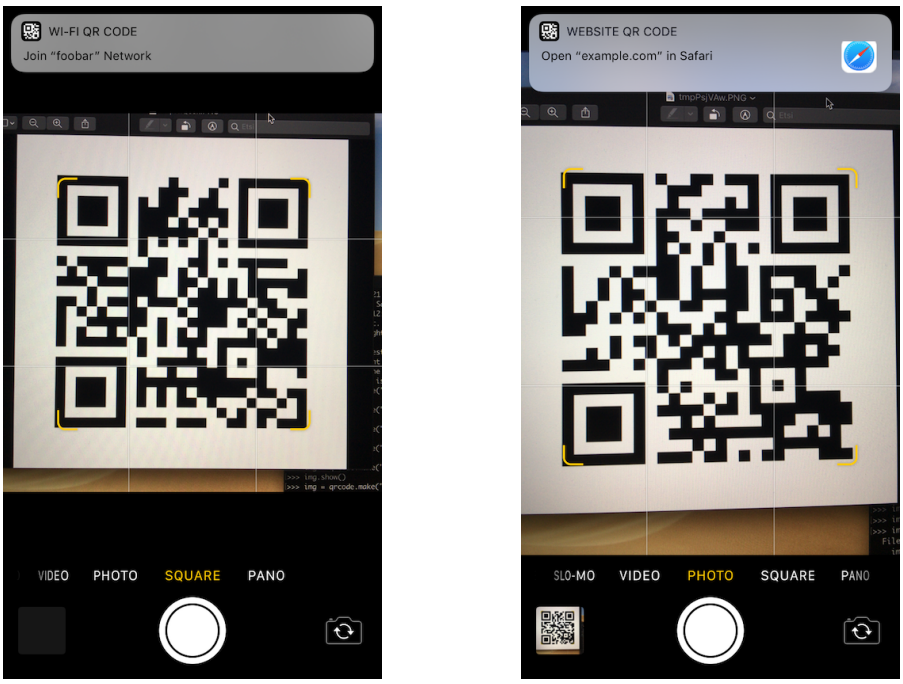
However, there are multiple other schemes, such as calendar events, contact information and credit transfer forms. For example, the Federation of Finnish Financial Services have released guidelines[1] how credit transfers are formatted as QR codes.

4.2.1 Default QR reader behavior

Most modern smart phones are capable in reading QR codes with either their default camera application or with third-party applications available from platform specific app stores. The typical process of scanning a QR code requires the user to point the smartphone camera at the QR code while the QR code reader application is running. When the QR code is detected, it is then decoded and often opened with a related application. More sophisticated QR code reader applications show the content inside the code prior to performing any action such as opening a URL or connecting to a wireless network. In iOS, the default QR reader needs to be enabled in the camera

[1]Finanssiala. Guidelines for the use of QR Code in Credit Transfer Forms. https://www.finanssiala.fi/maksujenvalitys/dokumentit/QR_code_in_credit_transfer_form.pdf

settings. When the option is enabled, the iOS default camera application scans for QR codes. If a QR code is detected, it is displayed as a notification banner to the user and requires user action. This banner also includes additional information about the nature of the QR code (e.g., text, Wi-Fi access or contact information). The iOS default QR reader, i.e., camera application is shown in Figure 4.8 with different notification banners for detecting QR codes encoded with (a) Wi-Fi access and (b) URL.



(a) iOS default QR code reader behavior for Wi-Fi network access.

(b) iOS default QR code reader behavior for URL.

Figure 4.8: Screen capture of iOS default QR code reader interpreting different QR codes.

On the Android platform, the process is slightly different due to the number of different device manufacturers and varying camera applications. Multiple Android smartphone manufacturers implement their own software and their camera applications do not always support QR codes. However, Android application stores, such as Google Play Store, Samsung Galaxy Store, and Amazon Appstore, allow users to download QR code readers from various application developers.

On Samsung Galaxy S8, QR codes can be read with Bixby Vision[2], which comes pre-installed on the phone. It is a part of Samsung's implementation of a digital assistant called Bixby[3]. It is similar to many digital assistants, such as Siri[4], Cortana[5], Google assistant[6], and Amazon Alexa[7], which are designed to perform various tasks with voice commands. The Bixby vision is launched from the default camera application or from designated hardware button on the side of the device. However, unlike the default iOS reader, it opens links and contact information without any further user action.

Figure 4.9: QR code with null separated data

Neither iOS nor Samsung default QR code applications read past a null character, which is interpreted as a terminator [39]. This means that these applications do not read all of the data in a QR code if there is a null character in the middle of a string. However, there are other applications that do not interpret the null as a terminator but instead as a character. While it does not follow the QR code standard, the null character could be used as a delimiter to fit multiple URLs within one QR code. Figure 4.9

[2]Bixby Vision. https://www.samsung.com/global/galaxy/apps/bixby/vision/
[3]Bixby. https://www.samsung.com/global/galaxy/apps/bixby
[4]Siri. https://www.apple.com/siri/
[5]Cortana. https://www.microsoft.com/en-us/cortana
[6]Google assistant. https://assistant.google.com
[7]Amazon Alexa. https://developer.amazon.com/alexa

displays a QR code encoded with the Python library package qrcode[8] with two URLs and strings of text. Each string is separated using \0, which both iOS and Samsung QR code readers interpret as a terminator. Most readers are able to read only the first entry, while third-party applications, such as Barcode Scanner[9], are able to decode the complete QR code.

4.2.2 Vulnerabilities in QR codes

There are two main methods for exploiting QR codes: either replacing the entire QR code or modifying parts of the QR code [40, 42]. For example, when QR codes are used in public information banners and advertisements, an attacker may replace the original QR code with a malicious QR code, e.g., in the form of a sticker. This is problematic in QR code reader applications, such as Samsung Bixby Vision, which process the code immediately without prompting for user action or displaying some identifiable information beforehand. Since the user has no way of knowing what is encoded within the QR code, the user might end up on a phishing website that mimics the appearance of a legitimate website, or to other malicious or disturbing websites. According to Krombholz et al. [42], many users are unable to determine the trustworthiness of a QR code, let alone the decoded URLs.

The other QR code exploit involves modifcation of a QR code. This modification attack focuses on the individual modules of the QR code, e.g., coloring white modules black [40, 42]. Kieseberg et al. [40] describe multiple approaches for modifying QR codes by attacking different parts, such as masks, character encoding, character count indicator, error correction or mixing modes.

These exploits can be used to perform phishing, fraud, attacks against reader software, social engineering, or attack machines that read these QR codes [40]. For instance, earlier versions of Android allowed QR codes to execute Man-Machine-Interface (MMI) codes [13]. This allowed QR codes with the tel:MMI URI format to factory reset Samsung devices simply by scanning them.

[8]Python QR code image generator library. https://pypi.org/project/qrcode/
[9]Barcode Scanner. https://play.google.com/store/apps/details?id=com.google.zxing.client.android

4.3 Data over sound

Sharing secrets in conversations face to face is considered to have strong authentication and integrity as an OOB channel [37]. In this type of human-interactive security protocol (HISP), all parties included in the conversation have a strong guarantee that their conversation is not being modified. Furthermore, they have visual confirmation of each other's identity; however, they may lack the affirmation of their conversation not being eavesdropped [37]. In this scenario, the participans could use the OOB channel (conversation) to verify the messages sent over the in-band communication channel that may use public key cryptography [37].

Similar to information exchanged in a face-to-face conversation, applications could use sound for the OOB channel. Computers are able to process sound quickly. This requires the computer to have connected speaker and a microphone, which are then employed in exchanging data over sound.

Madhavapeddy et al. [47] studied sound as a means of transferring and exchanging data including URLs and IP addresses. The results showed that sound is a viable option for transmitting URLs. However, the research also showed that transmitting long URLs over a low-bit-rate audio channel of 16 bps took too long to be considered a viable option for complex dynamic web applications with numerous HTTP GET requests. Moreover, Gerasimov & Bender [26] investigated various audio protocols for transmitting data. The paper includes a study of how disturbing people found the protocols. The results showed that the people preferred inaudible sound over audible sound.

The frequency range of human hearing is between 20 Hz and 20 kHz [64]. However, age is a key factor and affects the upper limit of human hearing. According to Rossing [64], only young people are able to hear frequencies up to 20 kHz frequencies, while middle-aged adults have an upper limit between 8 kHz and 15 kHz. Frequencies above 20 kHz are ultrasound and inaudible to most people. Nonetheless, decibel (dB) scale is an important factor in how well the sound is sensed [26].

According to Gerasimov & Bender [26], audible and inaudible sound can be safely employed in short-range scenarios, such as acoustic remote control or data transfer between devices, while people are in control of the data exchange. However, long-range data exchange is safe only when people are not in close vicinity. With long-range data exchange, the relatively slow speed of sound becomes an issue. Furthermore, the paper showed that standard

44.1 kHz digital-to-analog converter had difficulties in transmitting data over 18.4 kHz frequency. Therefore, not all devices are able to produce frequencies required for ultrasound. Nevertheless, the frequency range of human hearing and the nature of the sound should be taken into account when using sound as a channel for data. If the sound is disturbing to users, they might not want to use the technology. Furthermore, using too loud signal is harmful to humans as well as animals [64].

Currently, there exists no universal standards regarding data-over-sound methods. However, capabilities to interpret and generate these audio-based data-transfer methods can be implemented in application software.

4.3.1 Vulnerabilities in sound based channels

Due to the nature of the audio-based channel, it has vulnerabilities similar to other wireless systems. Therefore, audio-based channels are vulnerable to eavesdropping, spoofing, denial of service and man-in-the-middle attacks. These attacks are discussed in Chapter 4.1.2.

However, some of these attacks can be mitigated by using audible sound instead of ultrasound. For instance, the user is able to notice if other nearby devices are sending similar audio signals to disrupt or deny the communication. Furthermore, the user can identify the source of the signal based on the origin of the sound. To monitor these types of attacks in inaudible channels, user would have to rely on additional equipment.

Chapter 5

OOB Channel Implementation

In this chapter we describe the implementation of three different OOB channels. We begin by describing the tools used in this process, followed by the implementation of NFC, QR code and audio channels. Finally, we describe the development process of the Android application.

5.1 Tools and setup

Android Studio[1] is an integrated development environment (IDE) that provides various software development tools including an Android emulator. This tool was used in developing and testing the Android application during the implementation process. Furthermore, a laptop computer is used for simulating the peer device behavior, i.e., for generating and sending OOB messages. The laptop is equipped with an NFC card reader, speakers and a display. The messages are encoded as URLs and transmitted over NFC, QR codes and sound. While NFC and QR codes are supported by Android devices, we developed an Android application to interpret the audio channel with support for NFC and QR codes.

The Android application and the software for the NFC reader are written in the Java programming language while the QR code and audio channel are implemented with Python.

[1]Android Studio. https://developer.android.com/studio

5.2 NFC card reader

ACR122U-A9 is a contactless smart card reader manufactured by Advanced Card Systems Ltd. It follows both the ISO 14443 and ISO 18092 standards and supports MIFARE and ISO 14443 Type A and B cards, as well as FeliCa and NFC tags [1]. It is equipped with NXP Semiconductor's PN531 NFC controller chip, which in addition to the basic read-write and peer-to-peer modes allows the reader to enter card emulation mode via the TgInitAsTarget command and act as an NFC target [57]. However, the device does not have its own memory. This means that all operations and commands need to originate from the computer to which the device is connected via Universal Serial Bus (USB). In order to send NFC signals via the NFC card reader, the device needs its own native and Application Protocol Data Unit (APDU) commands, i.e., middleware commands. Fortunately, there exists a collection of Java libraries[2] for the NFC card reader for these commands.

Since the goal was interoperability with Android application, the NFC communication was chosen to use the Simple NDEF Exchange Protocol (SNEP), which is a peer-to-peer connection (Figure 5.1). It uses the NDEF format for transmitting data. SNEP is a stateless request and response protocol, which allows the confirmation of a successful message exchange. SNEP is supported by Android 4.0 (Ice Cream Sandwich) and later. This method is not compatible with older devices that account for 0.2% of all deployed Android devices [27]. Furthermore, as discussed in Chapter 4.1.2, even though the active mode is more vulnerable to eavesdropping, it is more robust against data modification.

For delivering the OOB message, i.e., URL from the IoT device to the user's mobile device, NDEF Message is formatted following the specifications discussed in Chapter 4.1. The URL is encoded as a well-known type (TNF value of 0x01) with full URI reference urn:nfc:wkt:U.

Multiple NDEF records included in the NDEF message allow flexibility. As we recall from Chapter 3.1, the EAP-NOOB protocol allows the peer to scan for available networks and, based on the scan, generate multiple OOB messages. In order to display these messages to the user, the peer device may include multiple NDEF records within an NDEF message. In this method, each URL is encoded as a separate NDEF record. Furthermore, if a new network is discovered later, it can be added to separate NDEF record within

[2]NFC tools. https://github.com/grundid/nfctools

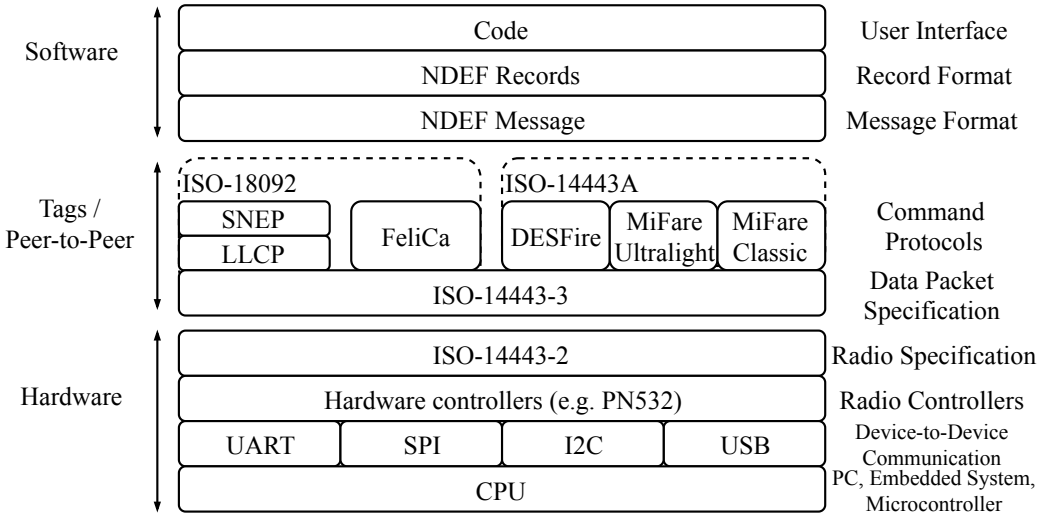


Figure 5.1: The NFC protocol stack [32].

the message. However, discovering added servers requires the user to perform the NFC exchange again.

Since the first record is always read on Android devices to determine how to interpret the NDEF message, we can take advantage of this by encoding it as a well-known text record (urn:nfc:wkt:T). This method circumvents the automatic behavior of opening URLs on most Android devices. This way, the user is able to access all of the following NDEF records within the NDEF message on default reader and choose which URL to open. Moreover, the text record can include identifying information regarding both the peer device as well as the available networks.

When OOB message is encoded as an URL, there are multiple ways for formatting it within the NDEF message:

1. Encoding each record as a URL. In this way, each scanned network is encoded as a separate record within the NDEF message. Most Android Internet browsers declare an NFC intent filter for URLs and as such, only process the first record by opening the link while discarding rest of the information.

2. Encoding the first record text record and consecutive records as URLs. In this method, the first text record includes information from the peer device. This could include information regarding the make and model of the device. Rest of the NDEF records are URLs.

3. Encoding all NDEF records as text records. This way each record is formatted as a text record. This allows additional information to be included within each record in addition to the URL. However, some Android devices may not identify included URLs as links, and therefore, they may not be clickable.

Since logging to the web page of the URL authenticates the device to the network and associates the peer device to the user account, it is essential for the user to identify the correct server. In the presence of multiple networks, the user needs to carefully choose the familiar server URL from the list. This may be a difficult task on Android devices that do not display the full server address. Therefore, in some scenarios it could be beneficial to include Android Application Record (AAR) within the NDEF message.

AAR can either prompt the smartphone to the Google Play Store or launch the application if it is already installed. This can be used to guide users to the application designed for the OOB channel. However, including an AAR within the NDEF message may be infuriating for users who do not wish to use designated applications or do not have access to the application store. Included AAR can be circumvented by utilizing third-party NFC reader applications. If need be, AAR is encoded as External (TNF = 0x04), with URI reference android.com:pkg and the application name as the payload.

5.3 QR codes

QR codes are generated with the Python qrcode[3] library package. In addition, Python Imaging Library (PIL)[4] is used for adding the name of the server into the image above the quiet zone (see Figure 4.7). This is used to help the user in the server identification process in scenarios where the peer device has scanned multiple available networks. The added human-readable URL gives the user some indication of the content and can be beneficial for scenarios where the user's QR code reader opens links immediately. Following the URI scheme, the QR codes start with prefix https://.

There are two ways for encoding URLs. Since, most of the default QR code readers do not support multiple sequential URLs, each URL can be

[3]qrcode 6.1. https://pypi.org/project/qrcode/
[4]Pillow 6.0.0. https://pypi.org/project/Pillow/

encoded as a separate QR code or each included URL is partitioned with a null character. However, the more characters are encoded in the QR code, the more complex the pattern becomes. More complex patterns require more time to be successfully decoded. In order to maintain support for default readers and to take advantage of more advanced readers, a number of URLs can be encoded into a one QR code. However, the first entry is needed to be cycled. In this method, advanced QR code readers can capture all of the information within the QR code, while standard readers can still read the first entry.

We implemented a Python program that generates periodically QR codes with null separated URLs. The first URL is cycled which allows the default readers to scan each entry while advanced QR code readers are able to read the complete QR code in one take.

5.4 Audio channel

Chirp.io[5] provides a software development kit (SDK) for multiple platforms and programming languages. In this thesis, we use the Chirp.io for transmitting the OOB message from a computer (Python Chirp SDK) to a smartphone (Android Chirp SDK). The proprietary protocol supports four acoustic transmission methods, shown in Table 5.1. These are labeled as standard, 16 kHz, 16 kHz-mono and ultrasonic. Each method has a maximum message length. The main differences between these methods are the sound frequency, message length and data transfer rates. In the standard audio transmission mode, the length of the message is 32 bytes, which is sent in a 4.52-second time frame. This results in a data rate of 56.6 bps. In the 16 kHz mode, the message length is limited to 90 bytes, which is sent in 8.16 seconds. This provides the fastest option with a data rate of 88.2 bps. For 16 kHz-mono option the data rate is 57.1 bps, while the ultrasonic option has the lowest transfer speed of 15.7 bps. In order to achieve faster data rates, the protocol supports multiple channels within the modes; however, this option requires special developer privileges.

The payload is encoded as an array of bytes, which the SDK transforms to audible sound. In the standard mode, the audible sound follows a melodic pattern. Each message follows a recognizable pattern consisting of an initial

[5]Chirp. https://chirp.io

Table 5.1: Supported acoustic transmission methods of proprietary Chirp protocol.

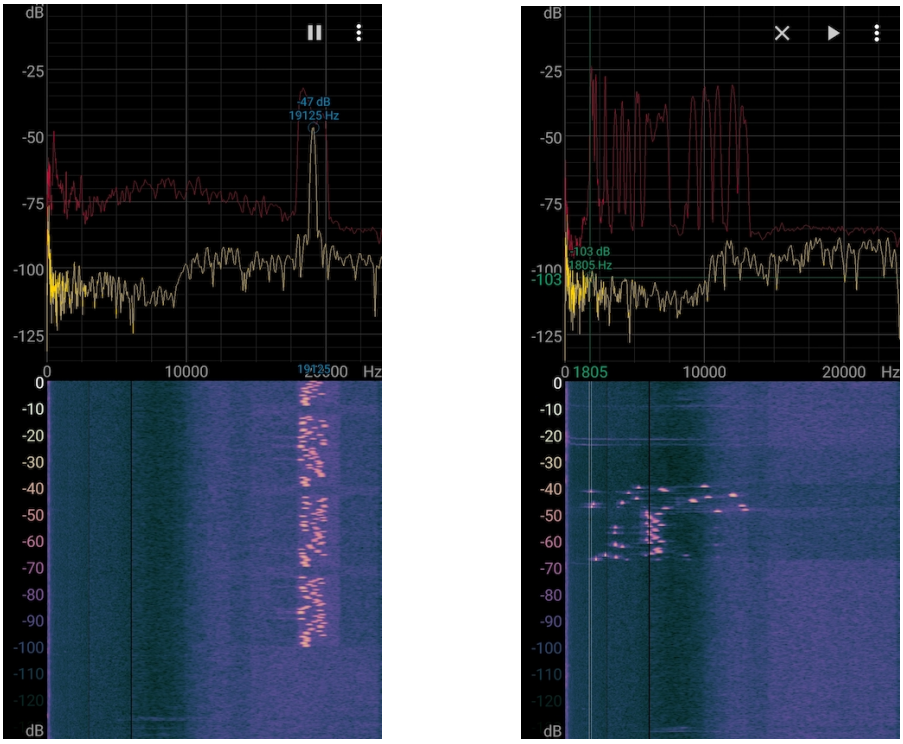
Mode	Max. message length	Transmission time	Data-rate
Standard	32 bytes	4.52 s	56.6 bps
16 kHz	90 bytes	8.16 s	88.2 bps
16 kHz-mono	32 bytes	4.48 s	57.1 bps
ultrasonic	8 bytes	4.08 s	15.7 bps

chirp pattern, i.e., handshake, the encoded data and the end of message pattern.

In testing runs of the proprietary Chirp protocol, the standard and ultrasonic options proved to be the most robust methods against noise (music or chatter did not disrupt message) while the 16 kHz option did not work on laptop speakers and required high-end speakers for the smartphone to intercept the signal. Furthermore, closer examination of the sound profile confirmed that the ultrasonic mode is not technically ultrasound since the operating frequencies are between 18 kHz and 20 kHz, as shown in Figure 5.2 (a). In the standard mode the frequency range is between 1.8 kHz and 13 kHz, shown in Figure 5.2 (b). Both figures display the same message that contains a URL. In addition, it displays the spectrogram visualization of the transmission of the 32 byte message. In the ultrasonic mode, the message is required to be split into 8 byte parts while the standard mode was able to transmit the message in one burst. The transmission was recorded with a Samsung Galaxy S8 with the Spectroid[6] application. Due to the standard mode providing reasonable data-rates, fairly good robustness against noice and being audible, it was chosen as the transfer method for the final implementation of the sound based channel between the computer and the smartphone.

As described in Chapter 3.1, the OOB messages of the EAP-NOOB protocol are relatively long when encoded as URLs. Therefore, most of the URLs need to be split into multiple messages. To indicate the end of a URL, the message is terminated with '+' sign, which is then removed from the final URL. The Chirp protocol could be used for two-way communication to take advantage of ACKs; however, this would significantly lengthen the duration of transferring the OOB message. It would also increase the time window

[6]Spectroid. https://play.google.com/store/apps/details?id=org.intoorbit.spectrum



(a) Chirp audio data transmission using ultrasonic mode.

(b) Chirp transmission with standard mode.

Figure 5.2: Screen capture of monitored audio graph of Chirp transmission with ultrasonic and standard setting.

for possible attacks. Furthermore, to indicate that the message is being received, user is displayed a loading bar. Moreover, this makes the wait time more pleasant for the user and keeps the user's attention [44]. In addition, each part of the message is rendered for the user as a visual confirmation that the message was received properly.

In order to combat data corruption and modification attacks, the audio channel is listened to only when the user presses a button, i.e., is prepared to perform the OOB step. The end-of-message sign stops the transmission.

5.5 Android application for EAP-NOOB

To avoid the identified problems with default NFC and QR code reader applications, we developed an Android application to examine the possibilities

for added security and usability. The application supports NFC, QR codes and audio OOB channels.

To take advantage of Android NFC tag dispatch system, the application is declared to filter for both HTTPS as well as plain text. Therefore, each time a NDEF tag is scanned, the Android operating system opens the application and the NDEF data is passed to the application requiring no additional scans.

The Application's initial view displays short instructions for navigating the application. The main view functions as a NFC reader. Moreover, the main view features a row of navigational buttons. These buttons navigate the user to the QR code reader, audio channel decoder or access options. The options button allows the user to enter HTTP authentication credentials and add servers to the list of trusted servers.

Android features a navigation bar, which includes navigations for back, home and overview. Since, most of the Android users are familiar with the navigation bar, the back button is utilized for navigating back to the main activity.

The application features a list of trusted servers, to which the user is able to add custom entries. This way, the user is notified if the scanned NFC tag, QR code or Chirp includes a URL of a trusted server. Furthermore, the application supports optional HTTP basic authentication as specified in RFC7617 [62] and RFC7235 [20]. It is a simple challenge and response method in which the server can request authentication information from the client. To take advantage of this feature, user needs to manually enter the authorization credentials. The token is sent over the HTTPS header if the scanned URL belongs to a trusted server. This allows the user to authenticate a device on a single tap requiring no user input. However, the requirements for singe-tap authentication are that the scan does not find other available trusted networks, the server is added as a trusted server and the user credentials are configured. The credentials are stored within the application's persistent storage, SharedPreferences[7].

Both authorization token and list of trusted servers take advantage of Android's SharedPreferences. Both the user credentials and the trusted server list are stored as key-value pairs in the XML files, located in the application's data directory. The user credentials is a string and stored with the key *token* while server list is stored as a string with the key *servers*. They are secured

[7]SharedPreferences. https://developer.android.com/guide/topics/data/data-storage.html#pref

by Android's file permission system. However, anyone with root access or the same application UID is able to access and modify them. Furthermore, the application does not allow cleartext and thus web pages without HTTPS do not work.

The developed application is capable of decoding QR codes. There are two methods for decoding QR codes, e.g., taking a picture and then processing the image or decoding the QR code from the camera feed. In this work, we implement the latter method because it decodes the QR codes immediately without requiring access to the device's storage. This way, the user is only required to click the authenticate button once the correct server is identified.

The application takes advantage of Google's Barcode API[8], which is a part of Google's Mobile Vision API. While supporting numerous barcode formats, the object detection is restricted to only QR codes. Initially, the scanning process was fairly slow due to the camera being out of focus most of the time. However, this was solved by utilizing the camera's autofocus capabilities, which expedited the scanning process significantly. To indicate a successful scan, the device vibrates. This notifies the user if the QR code is successfully decoded.

Android includes NFC data handlers in their Android framework API [28]. The application takes advantage of Android's NFC intent filter system. The application filters for ACTION_NDEF_DISCOVERED intents with HTTPS URI or TEXT. Therefore, each time a NDEF message's first record is a HTTPS URL or a plain text record, the application launches. If there are multiple applications that filter for this, the user needs to choose the application to handle the data. This is less intrusive approach compared to the AAR.

Android features Security Enhanced Linux (SELinux) to define boundaries for application sandboxing [3, 6]. Therefore, each application runs in a limited-access sandbox. If the application requires resources outside its own sandbox, it is required to requests for permissions [29]. In order to capture QR codes or sound bursts for audio channel, the application requests user's permissions to camera, and microphone. These permissions are classified as "dangerous" and therefore, explicitly require the user's permission [29]. Furthermore, the application requests permissions for accessing the Inter-

[8]Google Barcode API. https://developers.google.com/vision/android/barcodes-overview

net, NFC and vibration engine. However, these permissions are granted by the system automatically during installation because they are classified as "normal" [29].

Without being intrusive with the settings, e.g., denying access to the application without specific permission, the application functionality is only limited. That is, if the application is not allowed access to the camera then the QR reader does not work and informs the user that the camera is not enabled without pop ups or notifications. In addition, if NFC is disabled it only displays a text that the NFC is disabled without restricting access to the other application features.

As an experimental extra features, the application can be used for generating a QR code from a scanned NFC tag. This takes advantage of online QR code generator and therefore can not be considered a secure method for creating QR codes.

Chapter 6

Evaluation

In this section we report our findings from evaluating the advantages, limitations and vulnerabilities of each OOB channel against the requirements of EAP-NOOB. In addition, we discuss the usability of each OOB channel as well as the added benefits and vulnerabilities of the developed Android application. In the following subsections, the '−' sign indicates a negative feature and '+' sign indicates a positive feature.

When evaluating the security of a system, there are three goals that are most commonly considered. These are confidentiality, integrity and availability. Confidentiality means that secrets are protected. Integrity means that the information is not tampered with by unauthorized entities, while availability means the system is available at the required time. However, there are other aspects and goals that the CIA model does not cover, such as privacy or authorization, and which should be taken into account as well.

In the EAP-NOOB protocol, the primary goal of the OOB channel is to authenticate and verify the security of the in-band channel, i.e., guarantee the confidentiality and integrity of the primary channel. Furthermore, the authentication of the device is established with physical access to the device [7]. Therefore, the user should trust the authenticity of the OOB channel, and the user should trust the OOB messages. However, as we recall in Chapter 4, OOB channels have vulnerabilities where an attacker can interfere with the channels. Most of these vulnerabilities can be avoided by performing the device deployment in a restricted environment.

6.1 NFC

NFC provides a viable OOB channel for transferring multiple OOB messages at the same time. Android supports NFC by default and, it provides an intuitive way of authenticating a device by a simple tap.

The NDEF data format provides multiple ways for conveying URL encoded OOB messages. However, a potential weakness in the Android default NFC reader is that it does not display the full URLs when the NDEF record is encoded as well-known URI record. In most cases the information left out is the query parameters, such as the nonce and cryptographic fingerprint. For the most reliable method, OOB messages should be encoded as text records with additional information. This method circumvents the behavior of automatically opening URLs and provides information about the networks to the user so that the user can choose the correct network.

6.1.1 Usability

+ NFC follows the intuitive method of tapping the device, which is becoming a well-known method for device interaction. This can be seen, for example, in the growing trend of contactless payment [73]. For the passive mode, the maximum range for a successful tag read has been observed as 3 cm, while in the active mode, the data was successfully transmitted up to 8 cm.

+ NDEF enables multiple options for encoding the message. Applications can declare intent filters for associating with specific formats, and developers may include an AAR within the NDEF message for the Android system to either launch specific application or to prompt for the installation of the application from Google Play Store.

+ NFC provides a high-bandwidth channel, which can quickly transfer even relatively long OOB messages without saturating the channel capacity.

− NFC requires specific hardware and is not supported by all smartphones. Apple iPhones have been equipped with NFC modules limited only to contactless payment since iPhone 6; however, the latest iPhones are capable of reading NDEF tags without requiring third-party applications.

– The NFC channel is imperceivable to the human user, and the peer device needs to have some indicator, such as a LED light or a buzzer, to notify the user that it is ready to transmit the OOB message.

6.1.2 Security

– The card reader was found to show unexpected behavior when a contactless smart card was already present in the card reader. If an NFC tag (NXP MIFARE Classic 1k, NXP MIFARE DESFire EV1 or NXP MiFARE Plus) was already present on the ACR122U card reader, the smartphone would not detect the tag. Only after a SNEP command was issued, the smartphone registered the NDEF message of the NFC tag instead of the SNEP. After the NFC tag was removed, the reader would successfully transmit the OOB message. In some rare cases, the card reader would malfunction and the only way to get it working again was to remove the card reader from the USB port and re-attach it. At other times, the ACR122U card reader would get stuck in a loop where it tries to initiate the SNEP; however, the smartphone would detect the NFC tag instead. Two NFC tags would render the reader unable to transfer any messages.

  The described behavior allows phishing and misbinding attacks if the attacker has physical access to the peer device. For a phishing attack, the attacker needs to format an NFC tag with malicious URL that would, for example, mimic the appearance of a legitimate registration site. Thereafter, insert the NFC tag over the NFC terminal of the victim's device. This way the victim could be fooled into entering credentials or other sensitive information during the device setup process. A prudent user would likely notice the inserted tag but it could go unnoticed from an inexperienced user.

  In order to perform a misbinding attack, the attacker would first have to initiate the EAP-NOOB protocol on another device. Then, scan and copy the OOB message into an NFC tag. Thereafter, insert the tag over the peer device's NFC terminal. After the victim scans the device, the scan would reveal the OOB message of the attacker's device and thus the user could be fooled into associating it with their account. Now the attacker is in possession of the victim's device that may have access to various resources. However, this type of attack might be

quickly exposed since the victim would notice the user's own device would not indicate completion of the protocol. A prudent user would then revoke the unintended device association. In addition, the attacker would have to complete the attack within the expiration time of the OOB messages in order to perform a successful misbinding attack. The NFC tag containing an expired OOB message would thereafter act as a denial-of-service (DOS).

These types of attacks are targeted attacks and they require physical access to the peer device before the bootstrapping begins. Physical access means also that the attacker can include any serial number or other information displayed on the peer device into the tampered OOB message. However, as stated by Sethi et al. [67], most device-pairing protocols where authentication is established by a physical access are vulnerable to misbinding. The paper suggests a trusted path as one mitigation mechanism, such as LED light to indicate direct communication with the hardware.

The ACR122U card reader includes programmable led lights and a buzzer [2]. The LED light is red if there are no available NFC terminals and green if there is an NFC-capable device within the transmission range. Since these indicators are programmable they can not be considered as a trusted path [43]. However, they can expose the presence of the false NFC tag or sticker. Furthermore, since the SNEP protocol is a request response protocol, the card reader was able to detect the message not reaching the other end point. In this case, the smart card reader returned an error message instead of success message. Nonetheless, these mechanisms only help user detect the attack and do not prevent the user from scanning the malicious tag in the first place.

– NFC in the active mode is vulnerable to eavesdropping. The range for successful eavesdropping can be reduced significantly by switching to the passive mode. However, this would in return render the channel more vulnerable to data modification attacks.

6.2 QR code

QR codes provide a viable option for displaying OOB messages. However, in scenarios where there are multiple networks, the peer device's display plays

Table 6.1: QR code readability

PPI	QR code with 6 URLs	Single URL QR code
326	2.3 cm	1.1 cm
227	2.8 cm	2.1 cm
109	4.4 cm	2.5 cm

a significant role. It limits how many QR codes the device can output in a size that can be scanned reliably. If the device has a small display, it might have to cycle the QR codes. For devices that produce a printed QR code, the QR code size is also a factor that needs to be taken into account. As we recall from Chapter 4.2, both the length of the QR code and the error checking level contribute to the size of the QR code.

Multiple varying 126-character URLs were generated to evaluate the performance of QR codes. Single-URL QR codes smaller than 1.1 cm on a 326 Pixel-Per-Inch (PPI), 2.1 cm on a 227 PPI display, and 2.5 cm on a 109 PPI display were observed to be difficult for the default readers and third-party readers without digital zoom. However, iPhone's QR code reader with the manual digital zoom capability was able to decode even the smaller QR codes. Furthermore, it was noted that the error correction level does not affect the overall read speed of the QR code on the test devices. However, it might be a factor for devices with less powerful camera sensors and computational capabilities. With multiple URLs, the QR code becomes more complex and requires more area. This is shown in Table 6.1.

If multiple QR codes were displayed close to each other, all the readers displayed the contents of the first detected QR code and ignored the rest. This issue was less prominent in iPhone, since unlike the tested third-party readers, the iPhone default QR reader features a digital zoom, which narrows down the area of detection. Therefore, for peer devices that display the OOB messages as QR codes, it may be more reliable to display one QR code at a time.

Tampering a QR code on a display is unlikely to occur. In order to produce a tampered QR code on a display, the peer device needs to be already compromised. In this scenario, the user has no way of knowing that the device is compromised. Therefore, it should be recommended to reset the peer device before initiating the pairing process. Tampering of the QR code is also possible on printed surfaces. This would require the legitimate

user to leave the device unsupervised during the initial device deployment. To combat against tampering, Krombholz et al. [42] suggests use of complex color schemes. This causes the attack on QR codes to be more costly and makes it difficult to modify the QR code in an undetectable way.

6.2.1 Usability

+ QR codes are widely supported by smartphones due to large application stores. Furthermore, numerous URI schemes are supported by the Android and iOS mobile operating systems.

+ QR codes are fairly easy to use by simply focusing the camera towards the code. However, various studies have shown the scanning of QR codes to be a difficult task for users who are not familiar with them [67, 68].

− The QR code standard does not support multiple URLs and, therefore, displaying multiple URLs requires a separate QR code for each URL. This can result in a cumbersome scanning task in environments with numerous networks. In custom solutions, the QR code can be encoded with multiple URLs with custom delimiters. However, this eliminates the compatibility with most readers. In order to maintain some default reader support, it is possible to separate the URLs with the null character, which most readers interpret as the terminator. Default readers would be able to read the first URL while custom solutions would be able to read the complete QR code. However, this makes the QR code larger.

− Closely placed QR codes proved to be difficult to scan reliably. Neither default readers nor third-party solutions allowed the user to choose the QR code which to decode. Only the iPhone default QR code reader displayed a marker over the QR code which it had decoded.

− While it is possible to encode QR code with combined text and URL content, the URL is not clickable on default readers. Furthermore, only half of the tested third-party readers identified the link within the QR code.

6.2.2 Security

- Since QR codes are only machine readable, the human user cannot determine their content without a mobile device. This is problematic with devices that process the QR code content without user action. Therefore, it would be advised to include the URL in text form next to the QR code to indicate the content to the user.

- QR codes provide a visual OOB channel, and therefore, the implementation is vulnerable to opportunistic snooping attacks, such as shoulder surfing with a camera in public environments. Furthermore, QR codes can be spied over greater distances, e.g., by taking advantage of optical telescopes.

- Printed QR codes are vulnerable to tampering. This may occur in scenarios where the peer device outputs a printed QR code and the user leaves the printed QR code unsupervised. However, using color schemes increases the difficulty of unobtrusive tampering.

6.3 Sound

The sound channel provides a viable option in scenarios where there are only a few OOB messages to be sent. This is mainly due to the low bandwidth of the audio channel. Furthermore, since the relatively long OOB messages of EAP-NOOB must be split into multiple shorter messages, the Chirp handshake and end-of-message patterns add to the duration of the process. On average, a 126-character URL is successfully transferred in around 20 seconds. If the transfer is disrupted, the whole process must be started over again.

A long waiting period of over 15 seconds is often seen as detrimental to productivity and lowers user satisfaction [44]. However, the animated loading bar in addition to the updated status of the OOB message should significantly reduce the user's time estimation and therefore restlessness. Nonetheless, after the first OOB message, the user experience for utilizing the sound-based channel may become annoying due to the relatively long wait time.

While the transmitted data is not encrypted and can be easily eavesdropped, the channel is resilient against man-in-the-middle attacks. This is due to the fact that the transmitted data is authentic and can be verified by the user to originate from the device. Attempts to transmit audio data

over the original signal can be easily observed by listening, which exposes spoofing attacks against the process. The implemented audio channel is also resilient against data modification attacks. This is due to the fixed length of messages. Data insertion attacks results in corrupted data, which highlights the main vulnerability of the channel: the channel is easily disrupted. This can be achieved by playing louder noise at the same frequency. A gust of wind reaching the device's microphone during transmission will similarly disrupt the process.

The audio channel is vulnerable to data insertion in scenarios where the user only initiates the protocol on the smartphone and not on the peer device. However, in this scenario the user should be able to perceive that the peer device is not sending the signal.

6.3.1 Usability

+ Sound signal strength is easily controlled with the device's speaker volume.

+ Audio channel is suitable for smart devices with only speakers or microphone, in environments with only one or two networks.

− The audio channel has very low bandwidth. Therefore, transmitting relatively long OOB messages introduces a lengthy waiting period for the user. Waiting for a smartphone to receive an OOB message every 20 seconds and then requiring user action can be cumbersome to most users.

− Humans have varying thresholds for which frequencies are perceived as unpleasant. However, for close-range transmissions the audible signal does not need to be loud.

− The audio channel is unreliable in environments where the microphone picks up a lot of noise. Even a small breeze of wind to the microphone is enough to corrupt a message. Therefore, the audio channel is not viable for outdoor scenarios in windy conditions.

− In this implementation, the peer device requires user input to initiate the audio transmission since the application is not capable in sorting the message sequence.

6.3.2 Security

+ Sound frequencies used by the Chirp protocol are limited to rooms and the high-frequency audible signal does not travel through walls or windows.

+ Audible tune can be perceived and the user can confirm the signal is coming from the device. One of the problems in device-pairing protocols is the human imperceptibility of the wireless signal [41]. The audio channel approach mitigates the problem.

+ The sound protocol is immune to data modification attacks during active data transfers. Attempts to alter the data cause the messages to be corrupted.

− The channel can be easily disrupted. It is possible to disrupt the channel by playing similar frequencies as the Chirp protocol. This can be performed by simply recording the OOB channel for a brief moment and playing it back during the transmission.

− If ultrasound is used, the inaudible sound cannot be monitored, and the disruptive signals are hard to detect without tools.

− Similarly to NFC, the channel is vulnerable to eavesdropping. This would require the attacker to be in the same room or having an eavesdropping device in the room during the OOB message transfer.

6.4 Android application for EAP-NOOB

Overall, the application enhances the security and usability of the protocol. By filtering untrusted servers, it can prevent against most phishing and misbinding attacks. However, this requires the user to manually add trusted servers to the application. The added security results mostly from restraint. Usability is enhanced over default readers with the NFC intent filter, support for basic access authentication, and the list of trusted servers.

6.4.1 Usability

+ The application supports HTTP basic
authentication. If the user chooses to enter credentials, i.e., user name,

password and a trusted server name in the options menu, each initial HTTP request to the trusted server includes with the authorization header. Therefore, the user does not need to input the username and password each time they attempt to authenticate the smartphone to a trusted server. This eliminates the cumbersome step of having to manually enter the username and password. In the NFC-based OOB channel, this allows the user to immediately authenticate the IoT device to the server without any user action other than the initial scan with the smartphone. The only requirements for this are that the user's smartphone is unlocked and the peer device has successfully discovered the trusted server, i.e., generated the OOB message.

+ The trusted list can also be utilized for filtering networks. This improves usability in environments with multiple networks since the user does not have to manually browse through a list of URLs or server names.

+ The application supports one handed use since all the UI buttons are placed within the functional area of the thumb following the model specified by Bergstrom-Lehtovirta & Oulasvirta [9] for touchscreen surfaces.

+ Due to the NFC intent filter, the application is launched after scanning NDEF tags with HTTPS URI schemes or plain text and the data is displayed to the user. If the the scan finds a trusted server, the OOB message can be delivered immediately. In this way, peer device can be authenticated with a simple tap on an unlocked Android device.

− The Google's Barcode API does not support null-separated data. Therefore, the QR code reader can not decode complete QR codes that have null partitioned URLs.

6.4.2 Security

+ URLs are not automatically opened and the server URL is displayed to the user.

+ The application supports only the HTTPS and insecure HTTP links are discarded.

+ The application protects against phishing and IDN homograph attacks, i.e., the user is informed if the URL contains Cyrillic characters that are often used in said attacks. This is done by parsing through the URLs and checking if potentially compromising Cyrillic characters are found. However, it is important to note that this implementation is only a proof of concept and does include thorough character set checks. In order to protect against phishing, the application also supports a list of trusted servers. This allows the user to identify which scanned networks are trusted.

– While using third-party libraries, the application becomes vulnerable to supply chain attacks. The application developed for this thesis takes advantage of a proprietary third-party Chirp SDK, which allows the application to interpret the data transmitted over sound. Furthermore, the SDK sends anonymized analytical data back to the Chirp developers. According to the developers, the data is used to improve the service and no payload data is revealed.

– It is important to note that the Google Barcode API employs a powerful QR code detector as it can detect and process multiple barcodes in real time. This can lead to security problems where an attacker might be able to insert another QR code within the camera frame near the legitimate QR code. However, this problem exists also on other Android QR code readers. Only the iOS default QR code reader displays a marker over which QR code has been decoded.

Chapter 7

Discussion

In our findings, NFC is the fastest solution in relaying the OOB messages. This is mainly due to Android's native support for NFC and that it allows applications to declare intent filters. In environments where the peer device scans for multiple networks and generates multiple OOB messages, only NFC and QR code are viable options from the user experience perspective. The low bandwidth of the audio-based channel can be tedious for most users and is viable only in scenarios with one or two networks. The QR code is limited in regards to the capabilities of the peer device's display. In order to support default QR code readers, multiple URLs can be delimited with the null character. However, the implemented custom solution may not be as fast and reliable as the single-URL QR codes. The main benefit of the audio based channel is that the sound signal is perceivable by the user and can be identified originating from the device or if disruptive signals are used to interfere with the channel.

A challenge in the bootstrapping process arises with multiple networks and how to identify the correct network. The peer device is provided with some server metadata information in the initial contact with the server on the in-band channel, and some of it can be passed forward to the user with the OOB message. However, this information can be forged by malicious servers. The more OOB messages the user is required to browse through, the more work the user needs to perform. This can be problematic in environments with multiple networks. However, this is not a new issue, and RFC5113 [8] describes this issue in greater detail. Nonetheless, if the user is able to identify a familiar server URL, it should be safe to follow. After all, tampered nonces or cryptographic fingerprints result in EAP-Failure.

The user-assisted one-directional OOB channel is vulnerable to eavesdropping in both NFC and audio based solution while QR codes are vulnerable to snooping, e.g., shoulder surfing. These vulnerabilities can compromise the confidentiality of the OOB channel. However, the eavesdropping is a result of a targeted attack, and the attacker needs to know the physical location where the device bootstrapping is going to take place. While unlikely, it is possible to snoop the OOB message opportunistically as well. This may occur if the peer is deployed in public setting.

The NFC implementation is vulnerable to phishing and misbinding. However, these attacks require physical access to the device. When the NFC terminal is compromised physically, e.g., a sticker NFC tag placed over the NFC terminal, device reset does not resolve this issue. To combat this, the device needs to be able to notify the user when the interface is in trusted mode. As the EAP-NOOB [7] protocol draft suggests, a trusted hardware path between the user and the peer device would be needed. As proposed, for the peer device this could be a LED light. It could indicate if the peer is already configured or if the NFC terminal is in trusted mode.

While this thesis focused on URL-formatted OOB messages, with wide support for different types of URIs, NFC and QR codes can be used for constructing OOB messages with various formats. Example of this could be an SMS message. The SMS character limit can fit the PeerId, nonce and the cryptographic fingerprint whereas the host name can be replaced with a telephone number. An example of this is shown in Figure 7.1, which was generated with QRcode-monkey[1] tool. It implements a color scheme and a logo to render unobtrusive tampering difficult.

In initial deployment of devices with limited input and output, it is difficult for the user to detect if the peer device is compromised. Therefore, a device reset should be a part of the initial device deployment process. The device should have a trusted hardware button, i.e., a trusted path [7, 43], which returns the device to the unassociated state.

7.1 Future work

While this thesis examined OOB channels based on NFC, QR codes and audio for delivering URL formatted OOB message, it is possible to implement the

[1]QRCode-monkey. https://www.qrcode-monkey.com

Figure 7.1: A stylized QR code with a logo featuring SMS URI scheme

OOB channel using other methods and data formats as well. One method could be Visible Light Communication (VLC), which has recently gained attraction [36, 60]. VLC shows promise for IoT and automated smart-home scenarios and could prove a beneficial research topic with EAP-NOOB.

Additional studies on network discovery performance of the peer device and correct server identification in environments of multiple available networks could also help to further improve the EAP-NOOB protocol specification.

Chapter 8

Conclusions

In this thesis, we implemented and examined three OOB channels based on NFC, QR codes and sound for the EAP-NOOB types of bootstrapping protocols. Out of the examined methods, NFC and QR code were able to display multiple OOB messages without significantly affecting the usability. Since sound-based solutions are not supported by default, we implemented the audio channel using the proprietary Chirp protocol. The sound-based channel has low bandwidth and therefore is limited in terms of usability to scenarios where the peer generates only one or two OOB messages.

We examined the default NFC and QR code reader behavior on the Android mobile operating system. The examination showed that NDEF records formatted with URLs are automatically opened with the smartphone's Internet browser. For circumventing this unwanted behavior, we discovered that encoding of the first NDEF record as plain text eliminates the intent filters set by the Internet browsers. Our work also shows that URLs can be encoded as text records, which allows more information to be added to each OOB message.

While the QR code standard does not support multiple URLs, we implemented a flexible method which supports both standard QR code readers and more advanced readers. In this method, the peer encodes QR codes with null-partitioned URLs cycling the first entry. Therefore, advanced readers can decode the complete QR code after one scan while standard readers are able to decode the first entry on each scan. Simpler QR codes are more suited for scenarios where the capabilities of the peer device to display the QR code are a limiting factor.

All of the implemented OOB channels are vulnerable to spying, e.g.,

audio-based channel is vulnerable to eavesdropping, QR codes can be spied over great distances and, based on the literature, NFC in the active mode is vulnerable to eavesdropping from up to 10 meters away. This means that the initial device deployment may not be safely performed in public environments. However, the OOB messages have a configurable expiration time, which reduces the time frame for exploiting the snooped OOB message. In addition, we identified the NFC terminal vulnerability to phishing and misbinding attacks. These attacks require physical access or close proximity to the device.

We developed an Android application for EAP-NOOB, which can protect against most phishing attacks. The application can also help the user in selecting the correct network. The application can significantly reduce the extra work which is introduced by the user-assisted OOB channel. This is due to the ability to filter for servers, to cache user credentials and to configure the HTTP basic authentication.

In our implementations, it would be nearly impossible to detect a hijacked device. A device reset as part of the initial device bootstrapping process could resolve this issue. EAP-NOOB devices would benefit from a trusted path indicator, i.e., LED light or a buzzer to indicate if they are already deployed or if the OOB channel is active.

The performed literature survey shows that, overall, device bootstrapping is a multi-step process. In current commercial products, it often includes entering Wi-Fi credential followed by pairing the device to an auxiliary device, such as a smartphone, to establish ownership and secure registration. In this regard, EAP-NOOB can be a competitive as a alternative secure device bootstrapping protocol. We showed that the EAP-NOOB protocol allows the device bootstrapping process to be as simple as an NFC tap when combined with a designated mobile application.

Bibliography

[1] Advanced Card Systems Ltd. *ACR122U USB NFC Reader A Product Presentation*. ACS. https://www.acs.com.hk/download-manual/11/PPE-ACR122U-2.02.pdf.

[2] Advanced Card Systems Ltd. *ACR122U USB NFC Reader Application Programming Interface V2.04*. ACS. https://www.acs.com.hk/download-manual/419/API-ACR122U-2.04.pdf.

[3] Android Open Source Project. Security-Enhanced Linux in Android, 2019. `https://source.android.com/security/selinux`. Accessed 8 May 2019.

[4] Apple. Set up and use the Home app, 2019. `https://support.apple.com/en-us/HT204893`. Accessed 5 May 2019.

[5] Apple. Set up your Apple Watch, 2019. `https://support.apple.com/en-us/HT204505`. Accessed 5 May 2019.

[6] Asokan, N., Davi, L., Dmitrienko, A., Heuser, S., Kostiainen, K., Reshetova, E., and Sadeghi, A.-R. Mobile platform security. *Synthesis Lectures on Information Security, Privacy, & Trust 4*, 3 (2014), 1–108.

[7] Aura, T., and Sethi, M. Nimble out-of-band authentication for EAP (EAP-NOOB). Internet-Draft draft-aura-eap-noob-05, Internet Engineering Task Force, 2019. Work in Progress.

[8] Bari, F., Arkko, J., Aboba, B. D., and Korhonen, J. Network Discovery and Selection Problem. RFC 5113, Jan. 2008.

[9] Bergstrom-Lehtovirta, J., and Oulasvirta, A. Modeling the functional area of the thumb on mobile touchscreen surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2014), ACM, pp. 1991–2000.

[10] Berners-Lee, T., Fielding, R. T., and Masinter, L. M. Uniform Resource Identifier (URI): Generic Syntax. RFC 3986, 2005.

[11] Bluetooth SIG. Bluetooth Core Specification Version 5.1, 2019.

[12] Böhme, R., and Grossklags, J. The security cost of cheap user interaction. In *Proceedings of the 2011 New Security Paradigms Workshop* (2011), ACM, pp. 67–82.

[13] Borgaonkar, R. Dirty use of USSD codes in cellular networks. *TelcoSecDay, Heidelberg, Germany, March* (2013).

[14] Buttyan, L., Gligor, V., and Westhoff, D. *Security and Privacy in Ad-Hoc and Sensor Networks: Third European Workshop, ESAS 2006, Hamburg, Germany, September 20-21, 2006, Revised Selected Papers*, vol. 4357. Springer, 2007.

[15] Chattha, N. A. NFC — vulnerabilities and defense. In *2014 Conference on Information Assurance and Cyber Security (CIACS)* (2014), IEEE, pp. 35–38.

[16] Dhillon, G., Oliveira, T., Susarapu, S., and Caldeira, M. Deciding between information security and usability: Developing value based objectives. *Computers in Human Behavior 61* (2016), 656–666.

[17] Ericsson. Internet of things forecast, 2018. `https://www.ericsson.com/en/mobility-report/internet-of-things-forecast`. Accessed 5 March 2019.

[18] Felt, A. P., Egelman, S., Finifter, M., Akhawe, D., and Wagner, D. A. How to Ask for Permission. *HotSec 12* (2012), 7–7.

[19] Fidas, C. A., Voyiatzis, A. G., and Avouris, N. M. On the necessity of user-friendly CAPTCHA. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2011), ACM, pp. 2623–2626.

[20] Fielding, R. T., and Reschke, J. Hypertext Transfer Protocol (HTTP/1.1): Authentication. RFC 7235, 2014.

[21] Finkenzeller, K. *RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication*. John Wiley & Sons, 2010.

[22] Fischer, K., Gessner, J., and Fries, S. Secure identifiers and initial credential bootstrapping for IoT@ Work. In *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing* (2012), IEEE, pp. 781–786.

[23] Francis, L., Hancke, G., Mayes, K., and Markantonakis, K. Practical NFC peer-to-peer relay attack using mobile phones. In *International Workshop on Radio Frequency Identification: Security and Privacy Issues* (2010), Springer, pp. 35–49.

[24] Freed, N., and Borenstein, N. S. Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. RFC 2046, 1996.

[25] Funk, P., and Blake-Wilson, S. Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0). RFC 5281, 2008.

[26] Gerasimov, V., and Bender, W. Things that talk: using sound for device-to-device and device-to-human communication. *IBM Systems Journal 39*, 3.4 (2000), 530–546.

[27] Google. Distribution dashboard, 2019. `https://developer.android.com/about/dashboards`. Accessed 2 May 2019.

[28] Google. NFC Basics, 2019. `https://developer.android.com/guide/topics/connectivity/nfc/nfc`. Accessed 14 March 2019.

[29] Google Developers. Request App Permissions, 2019. `https://developer.android.com/training/permissions/requesting`. Accessed 1 March 2019.

[30] Grassi, P. A., Garcia, M., and Fenton, J. NIST Special Publication 800-63-3 Revision 3 Digital Identity Guidelines. *National Institute of Standards and Technology, Los Altos, CA* (2019).

[31] Haselsteiner, E., and Breitfuss, K. Security in Near Field Communication (NFC). In *Workshop on RFID security* (2006), pp. 12–14.

[32] Igoe, T., Coleman, D., and Jepson, B. *Beginning NFC: near field communication with Arduino, Android, and Phonegap.* " O'Reilly Media, Inc.", 2014.

[33] Intel. 802.1x overview and EAP types, 2019. `https://www.intel.com/content/www/us/en/support/articles/000006999/network-and-i-o/wireless-networking.html`. Accessed 2 February 2019.

[34] ISO/IEC 18004: 2000. Information Technology — Automatic Identification and Data Capture Techniques — Bar Code Symbology — QR Code 2000.

[35] ISO/IEC 18004: 2006. Information Technology — Automatic Identification and Data Capture Techniques — QR Code 2005 Bar Code Symbology Specification. ISO/IEC 18004:2006, 2006.

[36] Jovicic, A., Li, J., and Richardson, T. Visible light communication: opportunities, challenges and the path to market. *IEEE Communications Magazine 51*, 12 (2013), 26–32.

[37] Kainda, R., Flechais, I., and Roscoe, A. Usability and security of out-of-band channels in secure device pairing protocols. In *Proceedings of the 5th Symposium on Usable Privacy and Security* (2009), ACM, p. 11.

[38] Kainda, R., Flechais, I., and Roscoe, A. Security and usability: Analysis and evaluation. In *2010 International Conference on Availability, Reliability and Security* (2010), IEEE, pp. 275–282.

[39] Kentaro, F. Embedding Secret Data in QR Code. `https://fukuchi.org/works/qrhack/qrhack1.html`. Accessed 9 January 2019.

[40] Kieseberg, P., Leithner, M., Mulazzani, M., Munroe, L., Schrittwieser, S., Sinha, M., and Weippl, E. QR code security. In *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia* (2010), ACM, pp. 430–435.

[41] Kobsa, A., Sonawalla, R., Tsudik, G., Uzun, E., and Wang, Y. Serial hook-ups: a comparative usability study of secure device pairing methods. In *Proceedings of the 5th Symposium on Usable Privacy and Security* (2009), ACM, p. 10.

[42] Krombholz, K., Frühwirt, P., Kieseberg, P., Kapsalis, I., Huber, M., and Weippl, E. QR code security: A survey of attacks and challenges for usable security. In *International Conference on Human Aspects of Information Security, Privacy, and Trust* (2014), Springer, pp. 79–90.

[43] Latham, D. C. Department of Defense Trusted Computer System Evaluation Criteria. *Department of Defense Standard* (1985).

[44] Li, S., and Chen, C.-H. The effects of visual feedback designs on long wait time of mobile application user interface. *Interacting with Computers* (2019).

[45] Liu, B., Andersen, M. S., Schaub, F., Almuhimedi, H., Zhang, S. A., Sadeh, N., Agarwal, Y., and Acquisti, A. Follow my recommendations: A personalized privacy assistant for mobile app permissions. In *Twelfth Symposium on Usable Privacy and Security ({SOUPS} 2016)* (2016), pp. 27–41.

[46] Liu, Y., Yang, J., and Liu, M. Recognition of QR Code with mobile phones. In *Control and Decision Conference, 2008. CCDC 2008. Chinese* (2008), IEEE, pp. 203–206.

[47] Madhavapeddy, A., Scott, D., and Sharp, R. Context-aware computing with sound. In *International Conference on Ubiquitous Computing* (2003), Springer, pp. 315–332.

[48] Mayrhofer, R., and Gellersen, H. On the Security of Ultrasound as Out-of-band Channel. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International* (2007), IEEE, pp. 1–6.

[49] Mudugodu Seetarama, R. Secure Device Bootstrapping with the Nimble Out of Band Authentication Protocol. Master's thesis, Aalto University, 2017. http://urn.fi/URN:NBN:fi:aalto-201706135412.

[50] Naor, M., Rotem, L., and Segev, G. The security of lazy users in out-of-band authentication. In *Theory of Cryptography Conference* (2018), Springer, pp. 575–599.

[51] Nest Labs. Authorization Reference, 2019. `https://developers.nest.com/guides/api/authorization-reference`. Accessed 4 May 2019.

[52] Nest Labs. How Nest products use Bluetooth and NFC, 2019. `https://nest.com/support/article/Learn-how-your-Nest-products-connect-to-each-other-and-the-internet#bluetooth-nfc`. Accessed 10 April 2019.

[53] Nest Labs. How to set up a Works with Nest connection, 2019. `https://nest.com/support/article/How-to-set-up-a-Works-with-Nest-connection`. Accessed 4 May 2019.

[54] Nest Labs. OAuth 2.0 Authentication and Authorization, 2019. `https://developers.nest.com/guides/api/how-to-auth`. Accessed 4 May 2019.

[55] NFC Forum. NFC data exchange format (NDEF) technical specification.

[56] NFC Forum. What is NFC?, 2018. `https://nfc-forum.org/what-is-nfc/` Accessed 14 January 2019.

[57] NXP Semiconductors. *PN532 application note Rev. 01.00*. NXP, 2006. https://www.nxp.com/docs/en/nxp/application-notes/AN133910.pdf.

[58] Oracevic, A., Dilek, S., and Ozdemir, S. Security in internet of things: A survey. In *2017 International Symposium on Networks, Computers and Communications (ISNCC)* (2017), IEEE, pp. 1–6.

[59] Palekar, A., Josefsson, S., Simon, D., and Zorn, G. Protected EAP Protocol (PEAP) Version 2. Internet-Draft draft-josefsson-pppext-eap-tls-eap-10, Internet Engineering Task Force, 2004.

[60] Pathak, P. H., Feng, X., Hu, P., and Mohapatra, P. Visible light communication, networking, and sensing: A survey, potential

and challenges. *IEEE communications surveys & tutorials 17*, 4 (2015), 2047–2077.

[61] Peltonen, A. Formal Modelling and Verification of the EAP-NOOB Protocol. Master's thesis, Aalto University, 2018. http://urn.fi/URN:NBN:fi:aalto-201809034896.

[62] Reschke, J. The 'Basic' HTTP Authentication Scheme. RFC 7617, 2015.

[63] Reyes, A. R. L., Festijo, E. D., and Medina, R. P. Securing one time password (otp) for multi-factor out-of-band authentication through a 128-bit blowfish algorithm. *International Journal of Communication Networks and Information Security 10*, 1 (2018), 242–247.

[64] Rossing, T. *Springer handbook of acoustics*. Springer Science & Business Media, 2007.

[65] Rotem, L., and Segev, G. Out-of-band authentication in group messaging: Computational, statistical, optimal. In *Annual International Cryptology Conference* (2018), Springer, pp. 63–89.

[66] Sethi, M., Antikainen, M., and Aura, T. Commitment-based device pairing with synchronized drawing. In *2014 IEEE International Conference on Pervasive Computing and Communications (PerCom)* (2014), IEEE, pp. 181–189.

[67] Sethi, M., Peltonen, A., and Aura, T. Misbinding Attacks on Secure Device Pairing. *arXiv preprint arXiv:1902.07550* (2019).

[68] Shin, D.-H., Jung, J., and Chang, B.-H. The psychology behind QR codes: User experience perspective. *Computers in Human Behavior 28*, 4 (2012), 1417–1426.

[69] Siadati, H., Nguyen, T., Gupta, P., Jakobsson, M., and Memon, N. Mind your SMSes: Mitigating social engineering in second factor authentication. *Computers & Security 65* (2017), 14–28.

[70] Simon, D., Hurst, R., and Aboba, B. D. The EAP-TLS Authentication Protocol. RFC 5216, 2008.

[71] Soon, T. J. QR Code. *Synthesis Journal 2008* (2008), 59–78.

[72] Suomalainen, J., Valkonen, J., and Asokan, N. Security associations in personal networks: A comparative analysis. In *European Workshop on Security in Ad-hoc and Sensor Networks* (2007), Springer, pp. 43–57.

[73] Suomen Pankki. Payments statistics, 2018. `https://www.suomenpankki.fi/en/Statistics/payments-statistics/`. Accessed 9 May 2019.

[74] Telegram. End-to-End Encryption, Secret Chats. `https://core.telegram.org/api/end-to-end`. Accessed 6 May 2019.

[75] Telegram. Secret Chats. `https://core.telegram.org/blackberry/secretchats`. Accessed 6 May 2019.

[76] Thagadur Prakash, S. Enhancements to Secure Bootstrapping of Smart Appliances. Master's thesis, Aalto University, 2017. http://urn.fi/URN:NBN:fi:aalto-201709046881.

[77] Traficom Liikenne- ja viestintavirasto. TUPAS-tunnistamista kayttavilta asiointipalveluilta edellytetaan muutoksia. `https://legacy.viestintavirasto.fi/viestintavirasto/ajankohtaista/2018/tupas-tunnistamistakayttaviltaasiointipalveluiltaedellytetaanmuutoksia.html`. Accessed 6 May 2019.

[78] Victor, N. Enhancing the data capacity of QR codes by compressing the data before generation. *International Journal of Computer Applications 60*, 2 (2012), 0975–8887.

[79] Vollbrecht, J., Carlson, J. D., Blunk, L., Aboba, B., and Levkowetz, H. Extensible Authentication Protocol (EAP). RFC 3748, 2004.

[80] WhatsApp. WhatsApp Encryption Overview, Technical white paper, 2017. `https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf`.

[81] Wu, L., Du, X., Wang, W., and Lin, B. An out-of-band authentication scheme for internet of things using blockchain technology. In *2018 International Conference on Computing, Networking and Communications (ICNC)* (2018), IEEE, pp. 769–773.

Appendix A

List of tested Android QR code readers.

Bixby Vision by Samsung version 2.7.13.2

Barcode Scanner by ZXing Team version 4.7.8
https://play.google.com/store/apps/details?id=com.google.zxing.client.android

Lightning QR by Application4u version 2.0.3
https://play.google.com/store/apps/details?id=com.application
_4u.qrcode.barcode.scanner.reader.flashlight

QR Code Reader by TWMobile version 3.0.7
https://play.google.com/store/apps/details?id=tw.mobileapp.qrcode.banner

QR Scanner by Green Apple Studio, version 1.8.40
https://play.google.com/store/apps/details?id=com.apple.qrcode.reader

QR Scanner by EZ to Use, version 0.102
https://play.google.com/store/apps/details?id=app.qrcode